Workshop on Data Mining for Counter Terrorism and Security

May 3, 2003
Cathedral Hill Hotel
San Francisco, CA

To be Held in Conjunction with the
Third SIAM International Conference on Data Mining
(SDM 2003)

Theme Statement

The tragedy of September 11 had immeasurable and permanent effects on the United States and the rest of the world and brought issues of security and defense to the forefront. To help prevent such disasters, a successful security program would include provisions to secure borders, the transportation sector, and critical infrastructure. A critical enabler of such a program would be the ability to synthesize and analyze data from multiple sources.

The purpose of this workshop is to discuss ways in which data mining and machine learning can be used to analyze data from numerous sources of high-complexity for the purpose of preventing future terrorist activity. This is inherently a multidisciplinary activity, drawing from areas such as intelligence, international relations, and security methodology. From the data mining and machine-learning world this activity draws from scalable text mining, data fusion, data visualization, data warehousing methods. Papers in these areas with clear application to the issues of counter terrorism are particularly solicited.

Topics of interest include:

- Methods to integrate heterogeneous data sources, such as text, internet, video, audio, biometrics, and speech
- Scalable methods to warehouse disparate data sources
- Identifying trends in singular or group activities
- Pattern recognition for scene and person identification
- Data mining in the field of aviation security, port security, bio-security
- Data mining on the web for terrorist trend detection.

Program Committee

Dorothy Denning, Georgetown University
Jiawei Han, University of Illinois, Urbana-Champaign
John James, West Point
Anupam Joshi, University of Maryland, Baltimore County
Steve Kornguth, University of Texas, Austin
Rick Lawrence, IBM TJ Watson Laboratory
Nagiza Samatova, Oak Ridge National Laboratory
Ashok Srivastava (Chair), NASA Ames Research Center
Jeff Ullman, Stanford University

Organizing Committee

Ashok N. Srivastava (Chair), NASA Ames Research Center
Daniel Barbara, George Mason University
Hillol Kargupta, University of Maryland Baltimore County
Vipin Kumar, University of Minnesota and Army High Performance Computing Research Center

Keynote Speaker

Professor David Jensen, University of Massachusettes Amherst

David Jensen is Research Assistant Professor of Computer Science and Director of the Knowledge Discovery Laboratory at the University of Massachusetts Amherst. He received his Doctor of Science degree from Washington University in St. Louis in 1992. From 1991 to 1995, Dr. Jensen was an analyst with the Office of Technology Assessment, an agency of the United States Congress. While at OTA, he helped produce the first major assessment of data mining technologies for detecting money laundering. Dr. Jensen's current research focuses on relational knowledge discovery, with applications in intelligence analysis, web mining, and fraud detection. Dr. Jensen serves on program committees for the International Conference on Knowledge Discovery and Data Mining (KDD) and The International Conference on Machine Learning (ICML). He is a member of the American Association for Artificial Intelligence, The ACM Special Interest Group on Knowledge Discovery and Data Mining, and Computer Professionals for Social Responsibility.

Scalable Data Management Alternatives to Support Data Mining Heterogeneous Logs for Computer Network Security

William Yurcik * James Barlow † Yuanyan Zhou ‡ Hrishikesh Raje § Yifan Li ¶
Xiaoxin Yin ‖ Mike Haberman ** Dora Cai †† Duane Searsmith ‡‡

Abstract

Today no commercial technology exists for fusing audit log data from heterogeneous sources into a single framework for assessing suspicious computer network behavior. Identifying data relationships and patterns from heterogeneous sources on an instrumented network provides situational awareness and decreases the impact of accidental or malicious system failures that may have large impacts if they would appear. In this paper we compare different data management alternatives for accessing heterogeneous data sources by a data mining application and provide an overview of implementations under development at NCSA. We conclude that at this time there is not a consensus best alternative but each approach has significant tradeoffs that should become more accurately quantifiable as data mining projects for computer network security are implemented

Keywords: intrusion detection, data management, data warehouse

1 Introduction

Computer network security is a continuously escalating battle. Protectors harden operating systems, networks, and applications while attackers find vulnerabilities in any of these areas. However, there are four fundamental asymmetries that favor attackers in this battle: (1) an Internet connection provides worldwide access for all attackers directly to any machine, (2) while protectors must patch all vulnerabilities, an attacker must only find and exploit one vulnerability (all it takes is one unpatched vulnerability to bring down a system), (3) all it takes is one attacker to find and develop exploits for unpatched vulnerabilities since exploits are shared between attackers, and (4) the element of surprise - new undocumented attacks (ϕ days) are being continuously developed and used at opportune times and situations designed to maximize attack effectiveness.

Once this battle had low stakes, but since 9/11 the stakes have risen to include consideration of cyber-terrorists – terrorists who target underlying computer-based systems controlling critical infrastructures such as stock markets, electric power grid, various transportation systems, etc. While a *major* attack on a computer system supporting a critical infrastructure has not yet occurred[1], the possibility of human casualties and economic disruption makes this an important topic for study.

One counter-terrorism strategy against attacks on networked computer systems is to devise better detection tools. While it is difficult (and likely impossible) to harden any networked system against all attacks, most attacks do not occur instantaneously but rather have a definite sequence of events (from reconnaissance to system changes upon compromise) that can be identified and tracked resulting in either the prevention of an attack in real-time or reacting to an attack sooner after it has occurred. Attackers realize this and try to obfuscate their trail from intrusion detection systems (IDSs) with time (both very fast and very slow), deception, levels of indirection, etc. and in many cases this has been successful. The application of data mining to this problem is a higher-level attempt to discover semantic connections between attacker traces in computer network data that intrusion detection systems miss.

Commonly understood prerequisite infrastructure to facilitate data mining includes mass storage, processing power, data mining software, and a database management system. However, a last additional, often ignored, prerequisite is a scalable data management strategy. There has been little work focusing specif-

*National Center for Supercomputing Applications (NCSA)
†National Center for Supercomputing Applications (NCSA)
‡Dept. of Computer Science, Univ. of Illinois - UC
§Dept. of Computer Science, Univ. of Illinois - UC
¶Dept. of Computer Science, Univ. of Illinois - UC
‖Dept. of Computer Science, Univ. of Illinois - UC
**National Center for Supercomputing Applications (NCSA)
††National Center for Supercomputing Applications (NCSA)
‡‡National Center for Supercomputing Applications (NCSA)

[1] Isolated minor attacks have occurred such as distributed denial-of-service attacks against multiple companies, website defacements, and spam attacks but the authors do not feel these attacks rise to the level of a major attack.

ically on data management for data mining, research assumes data is available a priori to a data mining application and instead focuses on developing more effective data mining algorithms.[2] However, practical experience shows that data management for data mining can be almost intractable especially for a high volume application domain such as computer network security.

Audit logs used to monitor computer network security scale with the size of the instrumented network but are typically in the GB range per day for medium size organizations. The scope of the data management problem becomes clear when considering multiple audit logs remotely scattered across a network. There are several data management alternatives that may offer possible solutions but all have current technical drawbacks that must first be overcome.

In this paper we focus specifically on *scalable* data management for data mining in the domain of computer network security by comparing schemes from the literature and sharing practical implementation experience. The remainder of this paper is organized as follows: Section 2 reviews previous work in data management for IDSs. Section 3 summarizes audit logs for providing computer network security. Section 4 provides an overview of implementations under development at NCSA. We end with a summary and conclusions in Section 5.

2 Literature Review

In terms of data management, we distinguish three different states: (1) data collection, (2) data retrieval, and (3) data processing. For the purposes of this paper, data collection refers to network-based or host-based sensors generating audit logs to record activity, data retrieval traditionally refers to the virtual movement of log data to serve as input to a data mining application but we also use this term to encompass the virtual movement of output from remote data mining applications back to an operator, and data processing refers to the use of data mining algorithms to discover computer network attacks within logs.

IDS data management literature can be grouped into three general approaches: (1) centralized, (2) distributed, and (3) hierarchical. A centralized IDS focuses on processing all data at central depository. A distributed IDS focuses on the simultaneous (parallel) processing of data at multiple remote sources such that processing is loosely coupled (autonomous) with minimum data dependency [23]. A hierarchical IDS aggregates data at each of multiple layers in one direction from the raw source data up to the human operator with maximum data dependency (the input of each layer is directly dependent upon output of its adjacent lower layer).

Centralized approaches has traditionally relied upon centralized database systems but the only requirement is for all the data to be accessible for processing at a single virtual location – data can also be stored in sequential or specialized file formats outside of a database system.

In [1], the data is processed in a centralized fashion. The transaction information is extracted from the web servers, after which it is packed into some format to be sent to a separate host for further analysis. The same method is used in [18], where raw audit data is collected by sensors and shipped to the modeling engine. In [17], correlated logs from separate system components are fed to a database where data mining techniques are applied. Both systems described in [26] and [22] perform centralized operations on the audit logs aggregated from distributed data sources. A centralized approach for network traffic analysis is proposed in [8], where all traffic information is archived to a central host.

There is a hybrid variation to the centralized approach using a single layer of aggregation. Data is preprocessed remotely at one lower layer of sensors before the results (e. g., alerts) are delivered to a central place (e g., a database), where further operations are carried out. If more than one layer is involved then this hybrid variation of the centralized approach becomes what we define later as a hierarchical approach. Quite a few systems take advantage of this hybrid variation we will refer to as the *centralized-aggregation* approach. In [3], local processing is done at distributed sensors in network-centric IP fusion systems and resultant output data is sent to a host for fusion. The fusion system introduced in [10] consists of the IDSs that produce the alerts, and a database as a central repository where the alerts are stored. In [25], alerts generated by heterogeneous sensors are fused at a central location to produce meta alerts. In similar papers, data correlation is completed at a separate host where alerts coming from individual sensors are assembled [19, 9, 21].

According to [4, 24], research interest in distributed and hierarchical approaches has increased because they share workload across different nodes and thus improve the data management scalability of intrusion detection systems. Next we introduce representative approaches for both or these approaches.

Distributed approaches typically use autonomous agents which can be characterized by processing inde-

[2] a similar realization about the often ignored but prerequisite capability of data management has also occurred in the sister area of sensor networks where sensors are small, low-power wireless devices sensing the environment [12]

pendent of any centralized coordination and peer communication without any intermediaries. [13] presents one such representative distributed architecture that contains autonomous agents and no central repository. Each agent collects certain information from sets of hosts and it communicates interests (specifications of data it wants) to other agents. Different interests are sent to different agents or agent groups. If an agent gets data that is required by another agent, it sends data to that agent through a tree-like structure. And each agent generates alerts based on the data it has gathered.

[16] proposes a distributed approach for intrusion detection based on mobile agents. In this approach attack patterns are sent to mobile agents through a language called EQL. The agents detect local attacks, and communicate with each other to detect attacks involving multiple hosts.

[2] introduces several distributed approaches based on a publish-subscribe architecture. Each agent generates alerts from a single host and posts the alerts (or other information) at a central database. Each agent then retrieves information from the central database and makes actions.

The focus of hierarchical approaches is to decrease large volumes of data by using aggregation at multiple layers. This allows an operator to abstract away large volumes of data noise in order to facilitate discovering or investigating suspicious activity in the remaining data. Examples include aggregating the following groups of data noise: normal activity, false positive alarms, or multiple alarms from the same attack [6]. In this approach, the view from each layer is only of its next lower layer – there is no direct global view or direct view of the original source data. Unlike distributed approaches in which agent processing is autonomous, some hierarchical approaches use agents that form a multi-layer or tree-like structures but the processing is tightly coupled with layered data dependency.

AAFID [4] is a typical hierarchical approach. The AAFID architecture contains three layers of entities: agents, transceivers and monitors. Each agent monitors a certain aspect of a host. A transceiver is the external interface of a host that controls agents, processes data from them, and responds to commands issued by its monitor. Monitors are high-level entities that control transceivers from multiple hosts.

There are several other examples of hierarchical approach to data management for IDSs. Both [15] and [20] uses three-layer structures. The lowest layer collects information from sensors and may generates alarms for single hosts. The middle layer does correlation analysis or aggregation to generate high-level information, which is sent to the highest layer. The SHOMAR system uses a different hierarchical approach [24]. SHOMAR contains the following components: ID services, IDS managers, certificate authority and capability manager. An ID service is a sensor. Some ID services and an IDS manager form an ID cluster. The IDS manager collects information from every ID service in the cluster, performs data aggregation, and sends aggregated data to its parent. An IDS manager can be an ID service of another IDS manager such that IDS managers typically form tree-like structures. The other two components, the certificate authority and capability manager, control access to resources in the system. [11] uses a similar hierarchical architecture for data management.

2.1 Tradeoff Discussion Data management decision-making depends on project requirements, available resources (equipment, expertise), and time frame. However, there are general tradeoffs with each approach that we discuss in this section.

A centralized approach using a database to support data mining for heterogeneous logs has these advantages:

- effective data sharing: combining log files with individually different formats into a common uniform format. Users can also access the database either locally or remotely.
- efficient data access: database systems utilize a variety of sophisticated techniques to store and retrieve data efficiently. Such techniques include indexing, clustering, parallel processing, and query optimization.
- high reliability and stability: after years of refinement, database systems have become robust and easy to manage
- improving scalability: database systems have been used in many data intensive applications. A recent research article has reported a 5TB database by Oracle [14]

A distributed approach to support data mining for heterogeneous logs has these advantages:

- processing at a level closest to the raw source data which facilitates specialized algorithms at a desirable levels of resolution
- superior fault tolerance
- dramatically reduced processing (per individual machine) and network communication requirements (no concentrated congestion point)

- scalability to easily monitor new logs by adding corresponding new agents

A hierarchical approach to support data mining for heterogeneous logs has these advantages:

- flexibility in both number of levels and processing at each level
- centralized management of different layers due to data dependency
- reduced requirements for combining different raw data formats, can be combined gradually by layers
- large centralized mass storage requirements shifted to reduced and manageable distributed storage requirements at each layer
- scalability with increased processing at individual layers and/or increasing the number of levels

Table 1 compares the primary resources necessary to implement the different approaches: processing power in cycles for executing complex algorithms on large data sets, network capacity in bandwidth for near-real-time data retrieval, and large mass storage capability for queuing source data before (data collection) and after it is processed. There is one clear tradeoff: the centralized approach requires the most resources and the distributed approach requires the least resources with the hierarchical approaches in between.

Table 2 summarizes selected secondary criteria for comparison assuming each approach is feasible given the necessary resources. There are four clear tradeoffs: the centralized approach has a single-point-of-failure while the distributed approach can tolerant sensor/agent failures with minimal disruption, a uniform and complex data format is necessary for schemes with relatively concentrated processing, query flexibility is highest in centralized schemes and low in distributed/hierarchical schemes, and in terms of privacy centralized approaches represent higher risk as a concentrated target and larger compromise impact while the distributed approach presents multiple lower value targets with each having lower compromise impacts.

3 Heterogeneous Computer Network Audit Logs

A computer network contains a variety of different infrastructure devices each of which may be instrumented to produce audit logs. Though it is possible to perform simplistic signature matching on streaming network traffic in real time, it is impossible to analyze this data in real-time, in a period of minutes the data size becomes unmanageable. As a result most network analysis tools log the data for offline analysis. Although the topic of computer network audit logs is broad, a topic onto its own, we feel a brief introduction to some of the different types of logs is important in order to better understand the data management issues surrounding using these logs for IDSs or data mining.

We have made a point emphasizing data management for *heterogeneous* audit logs. The fact that the audit logs are different is significant because it promotes multiple views for attack discovery, robustness against attack, interoperability, extensibility, and flexibility. However, heterogeneity also eliminates possible data management efficiencies that may be possible from uniform record formats, uniform configurations, and uniform control.

It is important to note that computer network sensors generate streaming data and not batch log files. However, processing streaming data for data mining is an open research question beyond the scope of this paper (although briefly mentioned in Section 4.3). We create batch log files by collecting streaming data over defined time periods.

One of the most common ways of collecting computer network data into logs is the use of the **tcpdump** utility. This utility captures packet headers passing through a network interface set in promiscuous mode and displays binary traffic in a number of human-readable formats.[3]

While TCPdump is a valuable tool, it focuses on the TCP/IP suite of protocols. There are a large variety of other utilities for "sniffing" raw packets of any protocol from monitor points on a network. Referred to as "**sniffers**", the most effective programs are Ethereal and the Sniffer from Network Associates although there are many others.[4] As networks increasingly employ "switch" technology, sniffers that rely on a shared medium network (Ethernet) are being moved from end systems to servers and routers. Sniffer logs are uniquely valuable in discerning low-level attacks such as abnormal traffic attacks (e.g. fragmentation) however their scope is limited by their monitoring position within a network.

NetFlow logs contain records of unidirectional flows between computer ports across an instrumentation point on a network. These records can be exported from routers or software such as ARGUS or NTOP.[5] NetFlows are a rich source of information for traffic

[3] similar tools to tcpdump are Ipgrab and Iplog

[4] the word "sniffer" is a registered trademark of Network Associates

[5] ARGUS http://www.qosient.com/argus/
NTOP http://www.ntop.org

analysis consisting of some or all of the following depending on version and configuration: IP address pairs (source/destination), port pairs (source/destination), protocol (TCP/UDP), packets per second, timestamps (start/end and/or time duration), and byte counts (Note: there are different versions of NetFlow software that allow configuring for varying attribute resolution including sampling).

Syslogs are an industry standard for capturing information about networked devices by encoded messages by level (e.g. warning, error, emergencies) and by facility (e.g. service areas such as printing, Email, network). Syslog also functions as a distributed error manager by forwarding log entries to other machines for processing. In addition to pattern-matching syslog entries for known attack signatures, other examples of suspicious activity requiring further investigation include critical events (system reboots), unsuccessful login attempts, new account creation (especially with special privileges), connections from the same external host to many internals hosts on the same port (port scan), or cessation of logging messages from a host (may indicate the logging process has been deleted or a Trojan logging process installed).

Workstation logs are standard utilities that keep login/logout entries on a workstation's local hard disk (in addition to centrally maintained syslogs). Some application software also maintain access logs. One such example is the use of network-based "license servers". Workstation logs are a standard function provided on many operating systems but they are possible to disable. Each workstation also maintains a log of mail transactions originating from that workstation.

ARP cache at subnet routers and switches contain cached tables of recent conversions from IP addresses to physical hardware addresses for lookup efficiency. The entries are of two types: dynamic entries that are added/removed automatically over time and static entries which remain in the cache until the computer is restarted. Each dynamic ARP cache entry has a potential lifetime of between 2-10 minutes (depending on operating system settings, new entries are time-stamped) and a log of all entries can be created over a specified time period. The ARP cache is useful to determine static IP addresses; to identify unregistered/unknown, misregistered (including malicious spoofing), and misconfigured devices attached to a network; identifying what IP address(es) a particular hardware address is using; to debug if a particular device has connectivity; tracking unsuccessful connection attempts to devices that either are not currently on the network or do not exist; and lastly "arp cache poisoning" attacks against arp itself (the insertion of fabricated data). This log capability is becoming more important with the growth of wireless access points into networks.

Nameserver DNS cache contains mappings between fully-qualified hostnames and corresponding IP addresses (and corresponding name server hostnames and nameserver IP addresses) based on recent requests to other name servers. The amount of time a name server retains cache data is controlled by the time-to-live (TTL) for the data. These logs can be created via periodic snap shots of the cache timed shorter than the TTL to capture data before it expires. Host tables (. *rhosts* and *hosts.equiv*) that map IP addresses to hostnames also provides recent hostname-to-IP address mapping information. DNS cache is most effective in detecting IP/URL spoof attacks and malicious sniffing (by identifying machines performing high volumes of DNS queries with automated scripts).

Dial-up server logs maintain system accounting records on who makes ingoing/outgoing network connections to help identify suspicious activity ingoing or outgoing at this access choke point.

Kerberos logs contain all instances of the use of the kerberos authentication system in a network - Kerberos tickets requested. This information can be used to generate "login" graphs and determine who was logged into a particular workstation at a particular time.

SNMP logs , referred to as management information bases (MIBs), are databases of managed objects that store information about a wide variety of network device attributes. The SNMP (Simple Network Management Protocol) operator application involves monitoring network devices via polls to network device agents for specified MIB information or traps from network device agents notifying the operator of an event.

Routing table logs (e.g., inter-domain BGP, intra-domain OSPF or RIP) provide information about routing-based attacks ranging from: (1) individual misbehaving routers that drop/misroute packets or inject disruptively large routing tables, to (2) the systemic network-wide advertisement of false routing information or the instability caused by the propagation of worms. Global, local, or peer routing tables provide different vantage points for analysis.[6]

Firewall logs are important in a recursive way, to maintain the effectiveness of its internal rule set. A rule set exactly specifies what traffic to permit/block – typically growing in number of rules beyond human comprehension. A firewall is a computer or group of computers that interfaces between an internal network/computer

[6] one example of routing table logging is the Routeviews Project for BGP http://www.routeviews.org

and the Internet to enforce an organizational access control policy by processing packets/connections based on the rule set.[7] Firewalls can be used to monitor both normal activity (types of services requested and used, common external IP addresses accessing internal services, common access time patterns) and suspicious activity (probes to ports with no authorized services, external-to-internal flows with source internal IP addresses, outbound connections from uncharacteristic internal machines, and modification/disabling of the firewall rule set). Border routers with the ability to add static routes can serve a firewall function (NCSA does this)

Intrusion Detection System logs contain alerts indicating specific attacks. Generally, while a firewall has a proactive preventative focus, an IDS has a passive reactive focus. IDSs can be categorized in two ways: by sensor placement (network versus host) and by technique (signature versus anomaly), with all combinations producing logs. Real-time IDSs have been plagued by large log size, high false positive rates, and mimicry but incremental improvements are increasing their effectiveness for post-mortem forensics.

Mail logs maintain a log of completed transactions (as well as a queue of pending mail) including sender and recipient address, subject title, date and time of transmission, and size of file.[8] Common tests include: total length of time spent receiving and sending Email and the number of Emails by an entity (organization/group/individual) for period of time (day/week/month), stratify Email by time (work hours/off-hours) and common addresses, stratify size and type of file attachments, internal and external Email, and identifying dormant accounts.

Web server logs provide feedback on performance. Weblogs provide detailed records of requests to the webserver and statistical information about network traffic. The web log record attributes include: the source IP address from which a request was generated, whether a requests is satisfied, a userid determined by the HTTP authentication, a status code, and the size of the object returned with each satisfied request.

Dynamic Host Configuration Protocol (DHCP) server logs can be used to track unique IP address assignments to devices as they join/leave a network. DHCP servers manage two databases: (1) an Address Pool database for holding IP addresses and other network configurations and (2) a Binding database for mappings between Ethernet addresses and an entry in the Address Pool. Though best known for assigning dynamic IP addresses, DHCP can also assign a dedicated static address for a device that re-joins. On a network that uses DHCP with dynamic addresses, maintaining a log is absolutely necessary to be able to forensically associate dynamically changing IP addresses to specific devices/interfaces.

Scanning logs for defensive purposes are used to perform risk management by tracking vulnerabilities and notifying system administrators about potential exploits and patches that need to be installed. However, scanners are also the reconnaissance tool of choice for attackers to identify target IP addresses, servers, operating systems, and ports. Currently the four port scanning tools that stand out are nmap, nessus, SAINT, and the ISS scanner.

Other useful proprietary logs too specific to describe here include **operating system kernel logs**, **router traps,** and **application software logs**. Figure 1 shows a distribution of security relevant attributes across a selection of logs (although the individual attributes are too small to be readable). Some of the attributes may initially appear redundant since they are contained in multiple logs but this overlap is vital to enable event correlation between logs.

Figure 2 is a logical diagram of a centralized approach to the data management problem specific to the computer network security domain. Data in multiple types of logs may be retrieved by a centralized data mining application over the network. The time dimension is implicitly shown with multiple logs under each log type (NIDS is used as an example), with each log containing data from a defined time period.

4 NCSA SIFT Data Management Efforts

In August 2002, NCSA embarked on a data mining for computer network security project named SIFT (Security Incident Fusion Tools). The goal of the project is to fuse together many of the computer network audit logs we have identified for security event knowledge discovery.

As part of this effort we have already developed two breakthrough frameworks for visualizing individual logs across an entire IP address space (Class B address space = 65,000 computers with each computer having 65,000 different ports). Although we have organizational mass storage resources for data collection and high performance local/grid cluster resources for data processing, we have faced scalability challenges with data retrieval for fusing heterogeneous logs – moving data from data collection to data processing.

To appreciate the data retrieval for data mining challenge we face, here are approximate figures for just some of our log volumes:

[7] This includes two general classes of firewalls: (1) Internet/intranet firewalls (many variants) and (2) host-based firewalls

[8] An organization should have a clear Email security policy addressing privacy issues before data mining this content.

- NetFlow Logs - 200M - 4GB/day
- Syslogs - 250 MB/day
- IDS logs - 1 GB/day

These log sizes come with a caveat that they vary widely in size over time and are unique to our special NCSA environment. The point is a comparison of the log sizes relative to *current* data retrieval capabilities (acknowledging data retrieval capabilities are temporal and will change over time as technology evolves).

After surveying related work on data management for data mining and not finding a consensus solution, we embarked on four complementary efforts (see Figure 3). These four efforts map roughly to the approaches we described in the literature review: (1) a centralized database approach, (2) a centralized-aggregation middleware approach (to access log files at their original location), (3) a centralized-aggregation intelligent database approach (using DataSpace servers), and (4) a distributed agent approach.

4.1 Centralized Data/Centralized Processing

Figure 4 gives an overview of an architecture that employs a centralized database for unifying all the log activities. It includes a model to correlate data from multiple sources as it is collected, and to store only the information relevant to the behavioral models in a database format. A log from a sensor corresponds to a table in the activity database, and an event is stored as a row in the corresponding table. Correlations between logs are expressed using relations. For example, suppose a system call event from operating system is $<$ *Time-Stamp, ProcessID, API-Name* $>$, and a network TCP packet event is $<$ *Time-Stamp, ProcessID, Packet-Size, TCP-Flag, Src, Dst* $>$. They are correlated to each other if they have the same *ProcessID* and similar *Time-Stamp* .

The primary advantage of this organization is that it can easily interface to the activity miner and generate detection models. The database organization also has the advantage of supporting various queries. For example, when an anomaly is detected from mining the system call activities, the IDS system can query the activity database to see if there is any abnormal behavior in the network activities in the corresponding time range or by the corresponding processes. The IDS system can also query any aggregated information such as: the number of times a system call is made, or the average network traffic, or the highest busty network traffic, or the time range most logins happen etc. Similar to other centralized system, this approach provides a simple architecture, which is easy to implement and manage comparing to the other approaches discussed in followed subsections.

However, scalability is one of the primary challenges in this architecture. If the system is running for a long time or on a large-scale network, the database size may increase to Giga or Tera-bytes. To reduce the disk and memory footprint of the activity database, it is necessary to compress old events and archive or remove obsolete events. Moreover, stream-based data mining techniques are necessary to analyze only recent activities to generate model and detect intrusions. Time and space overheads are another problem with this approach. Since everything needs to be logged into this centralized activity database, making it a bottleneck in heavy loads.

To address these challenges, the activity log database may require special DBMS management. Most existing database servers such as IBM DB2, Microsoft SQL Server and Oracle are tailored for on-line transaction processing applications or decision support systems, and thereby may not be suitable for managing activity logs. Activity logs consist of time sequence data. Each activity's contribution to the model or the rule decays with time (last year's activities are less important than this year's activities). Therefore, old activities can be archived to secondary storage to reduce the database size and improve mining performance.

4.2 Distributed Data/Centralized Processing

Leaving raw data in its native format and original location allows data to retain the highest resolution with all its attributes without summarization or other lossy compression to shrink size. This is especially important for computer network security which requires a flexibility to ask hard to anticipate "what if" questions and depend on rare events that may be lost via summarization or compression.

Additional advantages with not moving raw source data from where it was initially created and leaving it in its native format include: (1) eliminates intermediate redundant storage and data consistency problems, (2) data that is not needed for processing does not need to be moved or reformatted, and (3) the responsibilities for data collection and retrieval can be decoupled for organization or technical efficiencies. The disadvantages of keeping raw source data in its original location are: (1) decoupling collection and retrieval means that when data is eventually retrieved for data mining, detailed knowledge of storage logistics is required, (2) each new high-level query will require a new, potentially huge, data retrieval (caching does may not help with such large log sizes) and (3) there is less fault-tolerance due to lack of redundant intermediate storage and thus more

dependence on system backups.

4.2.1 Middleware Mediation When trying to incorporate large amounts of data, from multiple sources, for a processing task like data mining, it can be very difficult to get the data you need when you need it. One approach NCSA is taking uses "middleware" mediation as a means to access source data remaining at its point of origination [5]. We use the term middleware to describe a software layer in between applications and network resources, specifically storage. Conceptually, middleware acts a centralized manager mediating and disassociating data processing from data collection – facilitating uniform access to heterogeneous logs based on attributes rather than filenames or physical locations. This approach benefits data management in the following ways:

abstracting the data

Using a middleware approach to data access allows data collection details to be abstracted from the data processing task (data mining application) such that high-level requests are transparently translated to low-level file access operations. As shown in Figure 5, the application does not have direct access to the data types. To retrieve data, an operator or application makes a high-level query to the middleware program that then determines what data is required and where the data is located. The data mining application or user does not know where the source data is stored, how it is stored, or in what format it is stored.

modularization

The middleware approach can also create a modular method to data access such that audit logs in different forms can be flexibly added and removed. This is possible since there are a small number of ways to store source data on the data collection machines, whether it is flat files, databases, or some other method. This also enables the use of reusable components for accessing new data sources.

To be more specific on our efforts at NCSA, Figure 5 shows a high-level request from either the user or data mining application sent to middleware. The middleware then mediates the request in one of three ways: (1) the middleware locates and then retrieves the exact log file(s) based on filename schema, directory structure, and desired time period (log files are created per log type and time period), (2) the middleware redirects the query to a central database corresponding to the log type selected (each log type may have a different database), or (3) the query is sent to an intelligent database server, in our case a DataSpace server hosting a specific log, which accepts complex queries for file attributes across multiple log files via communications between servers (explained in more detail in the next section).

A similar middleware approach focusing on network performance is described in [7]. This paper presents the Cichlid visualization tool that contains: (1) applications specific code for sensors and (2) an application independent visualization engine. The internal data representation is not attached to a particular application – user requests go through middleware to find and retrieve the corresponding data set.

4.2.2 DataSpace Servers The underlying data management architecture should (A) extract only that data which is currently requested for mining, (B) minimize the time to deliver the data, (c) retrieve data transparent from the physical aspects of storage. As we have documented throughput the paper, it may not be advisable to store all data on a central server given the large volumes. This lead to a search for an efficient way to retrieve portions of a data set irrespective of size.

NCSA has decided to implement DataSpace servers - an infrastructure for creating a data web (see Figure 6). Each DataSpace server keeps metadata about its logs allowing Internet communication between servers. This functionality represents logs as database tables by splitting text into columns and tuples as defined by the metadata. Also of interest are: (1) universal correlation keys as primary indices for active querying, (2) the predictive scoring and update protocol (PSUP) for event-driven real-time scoring and model development, and (3) the DataSpace transfer protocol (DSTP) for data queries over the web (the semantics of the protocol involve the rules for querying files based on their metadata descriptors).

To summarize, DataSpace inherently supports distributed data mining in the following ways:[9]

- DataSpaces supports distributed data by supplying a metadata standard, a federation mechanism, and easy to setup server code (Java, C++) that can be deployed near the data. A client can readily retrieve a table view that consists of columns joined from multiple DataSpace servers.

- DataSpace servers can interface with flat files, databases, and other formats. This ability to serve binary/text data reduces the large overhead of a uniform database record format. Server code is also extensible to new data format handlers.

- DataSpaces supports specialized socket layers (parallel sockets, etc.) to facilitate high throughput data transfers.

[9] DataSpace – A Web for Data http://www.dataspaceweb.net/

- DataSpace also supports publishing logs from a large number of sources with desired data privacy preserved with access control lists.

4.3 Distributed Data/Distributed Processing

Given the amount of data that can be generated by processes such as network packet traces, attempting to manage data using a centralized approach may be computationally infeasible.Even for a small organization, the size of audit logs can become quite large. For large ISPs, where data collected can be hundreds of gigabytes a day, moving source data becomes infeasible. The major issue with centralized data collection is that as data is being collected, the collection point can easily become overwhelmed with respect to the required network bandwidth needed to move the data, as well as processor cycles, and I/O requirements of the host to move the data through the TCP/IP stack and onto the storage device. With enough data sources, it is unlikely that a single host will be able to keep up. Centralized processing may work if the node acting as the sole processing point acts upon aggregated or generalized data of reduced size.

With the distributed data paradigm, keeping data at the collection points requires no additional resources other than those already in place to collect the data. Distributed processing also enables real-time analysis at the lowest level closest to the source that would not be possible if data is moved away from its site of origination or centralized processing of logs.

Distributed processing of data at collection points is another logical choice. A variation of distributed processing is aggregating data and then moving a data summary to another node responsible for answering data queries. The primary disadvantage of a distributed approach is the administrative overhead necessary to coordinate tasks on various hosts.

For a distributed processing model, a framework is needed where one can specify how and when data is to be processed providing a base set of features that all logging processes require. It also allows both ad-hoc and canned/general processing on data as it arrives. The actual processing of data can be defined statically or dynamically – processing depends on predefined paths/actions and also on the contents of the data packets. For example, if a certain signature is matched, the data flow path can be switched and other processing agents can be alerted as well. In this manner, independent agents that manage a particular data source can work together to accomplish overall goals.

At NCSA as we collect NetFlow records we store the data locally in audit logs for forensic purposes and also process each flow record as it arrives and either send it to another location for further processing or generate an Email or web page based on profiles or signature matches. Using the same framework at all data collection and processing points helps to mitigate software maintenance problems.

Used in conjunction with the distributed collection paradigm, data is kept where it was generated (e.g. host-based logs) or first collected (e.g. syslogs). Data can then be processed in real-time at these points and meta events can either be sent to some centralized location that is collecting events from various data generators or it can be sent to another collector to affect how it will process subsequent incoming data.

5 Conclusions

In this paper we outline different data management approaches for data mining audit logs. The core problem is scalability since individual logs, and thus also the union of multiple logs, can be huge in size depending on the scope of a particular network.

Different organizations approach this same data management problem by leveraging the unique resources available to them such as mass storage, processing power, software developers, network bandwidth, and database capacity. Specifically in this paper we present the multiple approaches NCSA is currently developing for data management of its heterogeneous network logs for data mining. As exemplified in our use of multiple approaches, at this time there is no clear consensus on the best approach for large networks. However, future plans include quantitative measurements for the different data management approaches we are implementing.

6 Acknowledgments

The following members of the NCSA SIFT team made significant indirect contributions to this paper (alphabetically): Loretta Auvil, Ratna Bearavolu, Randy Butler, Ruth Aydt, David Clutter, Kiran Lakkaraju, Doru Marcusiu, Bobby Rariden, David Tcheng, and Michael Welge.

References

[1] M. Almgren and U. Lindqvist, *Application-Integrated Data Collection for Security Monitoring* , Recent Advances in Intrusion Detection (RAID), (2001).

[2] T. Bass, *The Federation of Critical Infrastructure Information via Publish and Subscribe Enabled Multisensor Data Fusion* , 5th Itnl. Conference on Information Fusion, (July 2002), pp. 1076-1083.

[3] T. Bass, *Multisensor Data Fusion for Next Generation Distributed Intrusion Detection Systems* , IRIS Nat. Symp. on Sensor & Data Fusion, Johns Hopkins University/Applied Physics Lab., (May 1999).

[4] J. Balasubramaniyan et al., *An Architecture for Intrusion Detection using Autonomous Agents,* Annual Computer Security Applications Conf., (1998).

[5] P. Bernstein, *Middleware: A Model for Distributed Services* , Comm. of the ACM 39, 2(February 1996), pp. 86-97.

[6] E. Bloedorn, et al., *Data Mining for Network Intrusion Detection: How to Get Started* , AFCEA Fed. Database Colloq. and Exposition, (2001).

[7] J. Brown, A. McGregor, and H-W. Braun, *Network Performance Visualization: Insight Through Animation,* Passive/Active Measurement Wksp., (2000).

[8] V. Corey, et al., *Network Forensics Analysis* , IEEE Internet Computing, (Nov/Dec 2002), pp. 60-66.

[9] F. Cuppens and A. Miege, *Alert Correlation in a Cooperative Intrusion Detection Framework* , IEEE Symp. on Security & Privacy, (2002).

[10] O. Dain, and R. Cunningham, *Fusing a Heterogeneous Alert Stream into Scenarios*, within Appl. of Data Mining in Comp. Sec. Kluwer, (2002), pp. 103-122.

[11] H. Debar and A. Wespi, *Aggregation and Correlation of Intrusion-Detection Alerts* , Recent Advances in Intrusion Detection (RAID), (2001).

[12] D. Ganeson, D. Estrin, and J. Heidemann, *DIMENSIONS: Why Do We Need a New Data Handling Architecture for Sensor Networks?* HotNets-I, Princeton University, (2002).

[13] R. Gopalakrishna and E. Spafford, *Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents*, Recent Advances in Intrusion Detection (RAID), 2001.

[14] A. Joch. *Cracking the Code of Life* , Oracle Magazine, (Jan/Feb 2003), pp. 36-42.

[15] C. Kruegel, T. Toth, and C. Kerer, *Decentralized Event Correlation for Intrusion Detection* , Intl. Conf. on Info. Security and Cryptology, (2001).

[16] C. Kruegel and T. Toth, *Applying Mobile Agent Technology to Intrusion Detection* , ICSE Workshop on Software Engineering & Mobility, (2001).

[17] Z. Li, et al., *The Case of Centralized Logging Database for Intrusion Detection* , submitted to Hot Topics in Operating Systems (HOTOS), (2003).

[18] W. Lee et al., *A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions* , RAID, (2000).

[19] P. Ning, Y. Cui, *An Intrusion Alert Correlator Based on Prerequisites of Intrusions* , Technical Report, TR-2002-01, North Carolina State University, Department of Computer Science, (Jan. 2002).

[20] P. Petrov, et al., *A Hierarchical Collective Agents Network for Real-Time Sensor Fusion and Decision Support* , AAI/KDD/UAI Joint Wksp. on Real-Time Decision Support & Diagnosis Systems, (2002).

[21] P. Porras, M Fong, and A. Valdes, *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation* , Recent Adv. in Intrusion Detection, (2002).

[22] C. Silvestro, *Intrusion Detection Systems and Log Correlation.* Masters Thesis, Politecnico di Milano, (2002).

[23] E. Spafford, D. Zamboni, *Data Collection Mechanisms for Intrusion Detection Systems* , Purdue University, CERIAS Tech. Report 2000-8, (2000).

[24] J. Undercoffer, F. Perich, and C. Nicholas, *SHOMAR: An Open Architecture for Distributed Intrusion Detection Services* , Tech. Report, University of Maryland Baltimore County, (Sept. 2002).

[25] A. Valdes, K. Skinner, *An Approach to Sensor Correlation* , RAID, (2001).

[26] G. Vigna, B. Cassell, and D. Fayram, *An Intrusion Detection System for Aglets* , International Conference on Mobile Agents (MA), (2002).

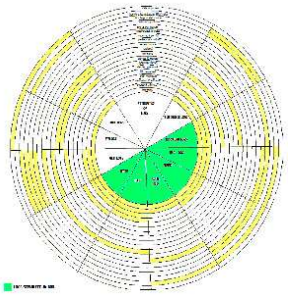

Figure 1: Attribute Distribution Across Audit Logs

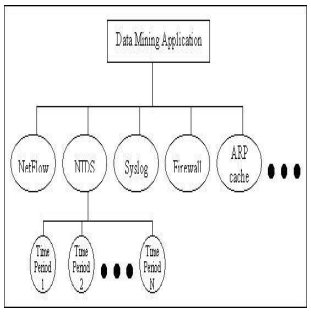

Figure 2: Management of Computer Network Logs

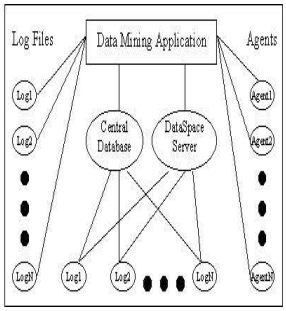

Figure 3: Parallel NCSA SIFT Data Management Efforts.

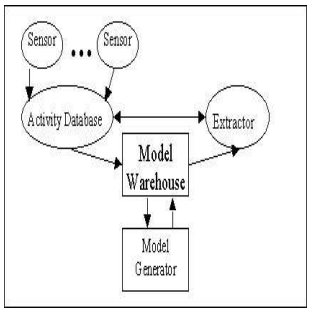

Figure 4: SIFT Central Database Architecture

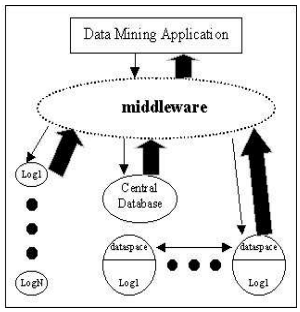

Figure 5: SIFT Middleware Architectures

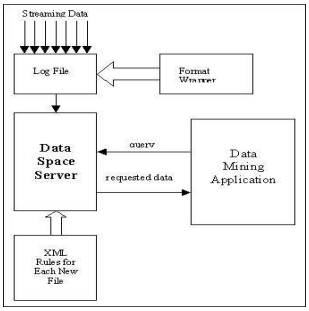

Figure 6: SIFT DataSpace Server Architecture

APPROACHES	processing power	network bandwith	mass storage
Centralized	high	high	high
Centralized - Aggregation	medium	medium	medium
Distributed	low/machine	high	low
Hierarchical	medium	low	medium

Table 1: Primary Comparison Criteria

APPROACHES	fault tolerance	data format	query flexibility	privacy: target/ scope
Centralized	low	complex	high	concentrated /global
Centralized - Aggregation	low-medium	medium-complex	medium	multiple-concentrated /limited-global
Distributed	medium-high	simple	low	multiple /limited
Hierarchical	low-medium	simple	low	multiple /global

Table 2: Secondary Comparison Criteria

Clusters Within Clusters: SVD and Counterterrorism

D.B. Skillicorn
School of Computing, Queen's University
Kingston Canada
skill@cs.queensu.ca

Abstract

We argue that one important aspect of terrorism detection is the ability to detect small-scale, local correlations against a background of large-scale, diffuse correlations. Singular value decomposition (SVD) maps variation, and hence correlation, into proximity in low-dimensional spaces. We show, using artificial datasets whose plausibility we argue for, that SVD is effective at detecting local correlation in this setting.

Keywords: singular value decomposition, attribute selection, correlation.

1 Introduction

Detecting terrorism can be posed as an unsupervised data mining problem in which the goal is to separate individuals into two classes, threats and non-threats. However, it is unusual because the members of one class (the threats) are actively trying to look as similar to members of the other class as possible. Without information about the particular data mining algorithm in use, the best strategy for doing this is to arrange for their attribute values to be modal.

This has two implications for data mining in terrorism detection: attributes should be such that it is hard to manipulate their values; and the data mining algorithms used should rely on the *relationships* between attributes, rather than simply their values. If attributes are chosen appropriately, then the activities of terrorists and terrorist groups may be visible as unexpected correlation, both among themselves and between the terrorists and their target. However, this correlation must be detected against a background of widespread diffuse correlation in the population at large.

Singular value decomposition is a useful tool for detecting unusual correlation because it transforms variation into proximity. Both distance measures and visual inspection can detect proximity far more easily than they can detect correlation directly.

We present preliminary results using artificial datasets. There is little experience to guide the form of such datasets, but we argue that the ones we use are at least plausible.

Results are encouraging, in the sense that SVD has high detection accuracy with reasonably low false positive rates. We are unable, so far, to specify an ideal algorithm schema for applying SVD, but we show that several strategies are effective.

2 Goals and assumptions

It seems implausible, given our present data mining technologies and understanding of the problems of counterterrorism, that data mining will be able to be deployed as a frontline tool against terrorism (at least in the immediate future). However, a useful role for data mining is as a filter, making it economic to select a manageable subset of individuals for further scrutiny using traditional intelligence techniques. In this view, the benefit of data mining is primarily to improve the effectiveness of other counterterrorism methodologies.

We assume, for the sake of concreteness, that we are dealing with datasets whose rows describe individuals and whose columns are attributes of those individuals. For example, datasets might contain information about which cities an individual has visited, or which flights he has taken.

Goals.

The untargeted case. Given a dataset, find clusters whose correlations are stronger than average, and select its members. If a group of terrorists are detectable as a cluster within the background of other clusters, then their target may also be detectable as part of the same cluster.

The targeted case. Given a dataset and a target, find clusters around the target whose correlations are stronger than expected. Any individual target might be expected to be part of multiple clusters representing his or her interests and collaborations. Hence, a cluster around them is likely to include parts of other clusters in which they are involved. This diffuse pattern should show an unusual concentration if there is a tightly-knit group that is focused on the target.

In the targeted case, the target is of the same kind

as the objects described by the rows of the dataset, individuals in this case. In the untargeted case, other kinds of targets are possible.

Assumptions. We wish to construct a detection model that will select some of these individuals for further scrutiny. We make the following assumptions:

- The potential consequences of failing to detect a terrorist are so great that a fairly high level of false positives is acceptable.
- Terrorists act in groups, so individual false negatives are acceptable provided that at least one member of a group is detected.

Attributes. The attributes in such datasets can be usefully divided into two kinds:

- *Incidental* attributes that describe properties and actions that are believed to be potentially correlated to terrorism. These may be static, such as country of citizenship, gender, income and so on; or based on actions such as purchasing particular kinds of plane tickets.
- *Intrinsic* attributes that describe properties and, more commonly, actions that are necessary to carry out a terrorist action, for example carrying out surveillance on a target site.

Both kinds of attributes have values that are shared by terrorists and the general population. The problem with incidental attributes is that if terrorists can learn the values of these attributes that trigger the detection model (and they can), then they can arrange to appear innocent. This leaves the detection model with a 100% false positive rate, which is the worst possible outcome.

The mechanism by which terrorists can learn the relationship of attributes to the detection model is by *probing*, the so-called Carnival Booth algorithm [4]. Terrorists arrange to be considered by the detection model while behaving innocently. Those who do not trigger the model can be reasonably certain that they will not trigger it again on subsequent, less innocent occasions. The use of incidental attributes is a major weakness of airline passenger profiling systems.

Models based on incidental attributes can be made more robust by adding uncertainty into the selection mechanism. This can be done by wrapping the detection model in a layer that obscures its precise functioning, for example, by randomly selecting some individuals who do not trigger the detection model and treating them as if they did. It can also be done by using families of detection models based on different thresholds for individual attributes (i.e. different discretizations of continuous data) or on different sets of attributes. These techniques all break the assumption that a person who has not been selected by the detection model on one occasion will not be selected on another occasion. However, these techniques all add expense and complexity; and using families of detection models risks one model failing to detect a threat that another model would have, which may be politically unacceptable if an incident takes place.

Intrinsic attributes are inherently better because terrorists are *forced* to have certain values for them. Of course, some of the general public will also share these values; but such attributes allow the set of individuals to be separated into those who are not terrorists and those who might be. As we have seen, incidental attributes do not do this reliably. Moreover, the set of individuals who can be eliminated will tend always to be much larger than the set of possible threats who remain.

The use of intrinsic attributes forces terrorists to come under scrutiny. Their only strategy then is to conceal themselves among that part of the population who share the same attribute values – but this becomes harder and harder as the number of attributes increases.

The power of intrinsic attributes can be seen in the aftermath of a terrorist action. Once such an action has taken place, the set of relevant attributes is clear – to plant a bomb in a certain place requires being in that place, for example. And once the correct set of attributes is known, terrorists are often detected very quickly. (This is also the basis of much police work – an alibi is a value for a very specific attribute which eliminates many possible perpetrators.)

Terrorists can only try to conceal their forced actions among those of many others. This is difficult for two reasons: (a) A terrorist group is forced to make coordinated actions, and such actions are potentially visible as correlations in the data. For example, if they meet to plan, then they are located at the same place at the same time. (b) A terrorist group is forced to carry out actions that are correlated with their target, and these actions are also potentially visible in the data. For example, they may travel the same route as the target but earlier in time.

These properties of datasets available for counterterrorism suggest that the problem is not closely related to outlier detection because terrorists try, as far as possible, to take on modal values for attributes. However, if intrinsic attributes are used, terrorist groups cannot avoid correlations both among themselves and with their targets. It is these correlations, which reveal themselves as locally dense regions within appropriate representations of the dataset, that data mining must search for – and which suggest the title of this paper. Some evidence for this is provided by Krebs [12], who

analyzed the connections among the group involved in the destruction of the World Trade Center. He showed that the members of the group were indeed tightly correlated. Of particular note is that a single meeting among a subset of them reduced the mean distance between members by 40% from its value given their relationships alone. Such is the power of intrinsic action attributes.

An immediate concern is that datasets describing any human population will be full of correlated subsets, and it might prove impossible to detect the correlations due to terrorism against such a noisy background. Consider the dentists of Pittsburgh[1]. We might expect that they would appear as a correlated group – they come from similar (educated) socioeconomic groups, they live in similar settings, and they travel to similar conferences and conventions. However, as we consider more aspects of their lives, these correlations begin to be diluted by others: they travel to differing parts of the country for family occasions, their children insist on holidays in different places, and they have different hobbies. The terrorists of Pittsburgh (should there be any) might also appear strongly correlated by a few attributes, but this correlation is much less likely to dilute as further attributes are considered.

More formally, the reasons why correlation in terrorist groups might be visible against a background of widespread correlation are these:

1. Most individuals are part of a fairly large number of subgroups with whom they are correlated – enough that the strength of membership in each one is quite small.

   Consider the folk theorem about six degrees of separation, the contention that a chain of acquaintances of length less than or equal to six can be built between any two people in some large population (originally the population of the U.S. in Milgram's original work, now often claimed for the total world population). If a given individual is acquainted with (say) a individuals, then each of these a individuals must be acquainted with a fairly large number of others outside the original set of a or else the powers do not increase quickly enough (since $a^6 \approx$ the large population).

   This result contradicts our intuition that an individual's social circle tends to be small. The resolution (see, for example, [14]) is that such small social circles are bridged by rare, but not too rare, 'long-distance' connections.

   Acquaintanceship is a reasonable, although not perfect, surrogate for correlation in the kind of datasets we are interested in – we would not be surprised that acquaintances would turn out to be fairly well correlated in large datasets – they live in similar places and have similar lifestyles, including travel arrangements. What is less obvious is that the 'long distance' connections in acquaintanceship are likely to produce strong correlations as well – for an acquaintanceship survives only if its members have 'something in common'. Hence the implication of six degrees of separation (and the existence of short paths in acquaintanceship graphs) is that correlation smears rapidly across subgroups because of the richness of cross-connections of common interests and behavior.

2. We might expect terrorists to be substantially less connected by correlation than most people because they have a much narrower focus. Informally, we might suspect that terrorists don't buy life insurance, don't take holidays, don't buy lottery tickets, and don't have children in Little League. We quote from Krebs [12, p49], relying on previous work on the social network structures of criminals: "Conspirators don't form many new ties outside of the network and often minimize the activation of existing ties inside the network".

These properties provide some assurance that a signature for terrorist actions exists in datasets that are sufficiently large and diverse. Note that, in this context, high dimensionality is a benefit because it acts to smear the background correlation in the population at large.

3 Data Generation Models

Since, for obvious reasons, real datasets containing terrorist actions are not available, the quality of detection models will have to be evaluated using artificial datasets. This immediately raises the question of what kinds of datasets are plausible.

Intrinsic attributes can be divided into those related to actions and those related to state. We now consider the properties of each.

For action attributes, an immediate issue is how to handle their temporal nature. They could be coded with time signatures attached and temporal data mining techniques used – but I am not aware of any present data mining technology powerful enough to detect temporal subsequences when different parts of them are carried out by different individuals (this is an interesting problem, though). It seems simpler, and perhaps more robust, to handle temporal properties by creating attributes for actions covering a period of time. For example, if visits to New York are an action of interest, then these can be converted into attributes as visits per

[1] Apologies to both dentists and Pittsburgh for this example.

month: January visits, February visits, and so on. It is also sensible to use overlapping time periods (creating partly correlated attributes) to avoid sensitivity to boundary choices.

Attributes representing actions will also be: (a) Sparse, because only a small fraction of the total population of individuals will carry out any given task (e.g. only a small fraction of the U.S. travelling public visit San Francisco in a given month). (b) Have a frequency distribution whose mode is close to 1 and which decreases quickly (e.g. those people who visit San Francisco in a given month mostly visit only once).

Such attributes can plausibly be generated by first introducing a high level of sparseness and then generating the nonzero values using a Poisson distribution with mean close to 1.

State attributes will have much flatter distributions. For example, the locations of residences of members of a terrorist group around a target might be expected to conform to a normal distribution because of the pressures for closeness to the target, counterbalanced by the pressure to remain far from each other. State attributes will also tend to be dense (everyone has to live somewhere).

4 Singular Value Decomposition

Singular Value Decomposition (SVD) [7] is a well-known technique for reducing the dimensionality of data.

Suppose that a dataset is represented as a matrix A with n rows (corresponding to individuals) and m columns (corresponding to their attributes). Then the matrix A can be expressed as

$$A = USV'$$

where U is an $n \times m$ orthogonal matrix, S is an $m \times m$ diagonal matrix whose r non-negative entries (where A has rank r) are in decreasing order, and V is an $m \times m$ orthogonal matrix. The superscript dash indicates matrix transpose. The diagonal entries of S are called the *singular values* of the matrix A.

One way to understand SVD is as an axis transformation to new orthogonal axes (represented by V), with stretching in each dimension specified by the values on the diagonal of S. The rows of U give the coordinates of each original row in the coordinate system of the new axes.

The useful property of SVD is that this transformation is such that the maximal variation among objects is captured in the first dimension, as much of the remaining variation as possible in the second dimension, and so on. Hence, truncating the matrices so that U_k is $n \times k$, S_k is $k \times k$ and V_k is $m \times k$ gives a representation for the dataset in a lower-dimensional space. Moreover, such a representation is the best possible with respect to both the Frobenius and L_2 norms.

SVD has often been used for dimensionality reduction in data mining. When m is large, Euclidean distance between objects, represented as points in m-dimensional space is badly behaved. Choosing some smaller value for k allows a faithful representation in which Euclidean distance is practical as a similarity metric. When $k = 2$ or 3, visualization is also possible.

Another way to understand SVD is the following: suppose that points corresponding to both rows and columns are plotted in the same k-dimensional space. Then each point corresponding to a row is at the weighted median of the positions of the points corresponding to the columns and, simultaneously, each point corresponding to a column is at the weighted median of the positions of the points corresponding to the rows. Hence SVD can be viewed as translating correlation or similarity into proximity. Unfortunately, only positive correlation is taken into account by SVD, so that rows that are strongly negatively correlated will not be placed close together in space.

SVD measures variation with respect to the origin, so it is usual to transform the matrix A so that the attributes have zero mean. If this is not done, the first singular vector represents the vector from the origin to the center of the data, and this information is not usually particularly useful. For example, when A is the adjacency matrix of a graph, it is the second singular vector which describes the partitioned structure (if any) of the graph.

While SVD is a workhorse of data manipulation, it has number of subtle properties that are not well-known. We will use four of them.

Fact 1: The singular value decomposition of a matrix is insensitive to the addition (or subtraction) of independent zero-mean random variables with bounded variance [1]. This property has been used to speed up the computation of SVD by sampling or by quantizing the values of the matrix. In counterterrorism, the effect we are looking for is so small and the results so important that neither of these is attractive. However, the fact does explain why SVD is good at detecting clusters within clusters – the outer cluster representing the majority of the data has zero mean (by normalization) and so, by the *fuzzy central limit theorem*, increasingly resembles a normal distribution as the number of ordinary individuals (and the number of attributes) increases.

Fact 2: SVD is a numerical technique, and so the magnitudes of the attribute values matter. However, multiplying the attribute values of a row of A by a scalar larger than 1 has the effect of moving the correspond-

ing point further from the origin. Because the positions of all of the other points depend, indirectly, on their correlations with the scaled point, via their mutual interactions with the attributes, points that are correlated with the scaled point are pulled towards it. When there is little structure in the low-dimensional representation of a dataset, this scaling technique can be used to find the individuals who are (positively) correlated with a given individual. In practice, this often makes it easier to see a cluster that would otherwise be hidden inside another in a visualization.

Fact 3: Although SVD translates only positive correlation to proximity, negative correlation information can be extracted from the SVD indirectly. Let A_k be the product

$$A_k \;=\; U_k S_k V_k'$$

The matrix $C = A_k A_k'$ can be understood as a kind of correlation matrix in which some kinds of correlation have been discarded (those arising from dimensions $k+1$ and higher) while some higher-order correlation information has been included [10, 11]. In other words, entries in C are non-zero even when the corresponding entry of AA' was zero (that is, even when there is no direct correlation between a pair of individuals).

The connection between the sign of entries in this matrix and correlation was noticed empirically by Konstotathis and Pottenger [11]. It is also related to a well-known technique for partitioning graphs using spectral methods [2, 9]. The following explanation shows why the magnitude of the entries of C can be regarded as correlations (both positive and negative) between individuals. Consider the ijth entry of C. This entry arises as a sum of values, each of which is the product of a column of U, an entry of S, and a row of V. Such an entry is negative when individuals i and j are on opposite sides of the origin in one of the dimensions. A sum of such values represents an average 'reflection' in the origin in all k dimensions.

When the ijth entry of C is negative, we can conclude that individuals i and j are negatively correlated. In fact, we can go further – when entry C_{ij} is smaller than C_{ii} we can conclude that the correlation between individuals i and j is weak.

Fact 4: The decomposition depends on all the data used, both normal and anomalous. The precise geometry of the detection boundary of SVD is hard to predict without performing the decomposition, and impossible without knowledge of the dataset. Hence, a terrorist group cannot reverse engineer the transformation to determine how they will appear, even knowing that SVD is being used. In particular, SVD is resistant to probing attacks since any attempt to probe cannot control for the innocent individuals considered at the same time.

5 Algorithms

There are a number of algorithmic tools based on SVD that can be combined in various ways. Our results do not indicate a clear optimal strategy for using SVD, but they do reveal several effective tactics.

If we start with a high-dimensional dataset (e.g. $m = 30$), then we can apply SVD, truncate U to two or three columns and plot the corresponding rows. Their positions are the best low-dimensional representation of the original data.

Zero-mean normally-distributed data appears as a spherical cluster centered on the origin even in low dimension. Any correlated set of individuals tends to appear as a cluster further from the origin along the first singular vector (and sometimes the second). Hence, a frequency plot of the first column of U may sometimes reveal a possible target and associated terrorists. Visual inspection of the plot can also be revealing.

When a target is known, there are several further options for selecting individuals as potential threats:

- Project points onto the vector from the origin to the target point in the transformed two- or three-dimensional space, and classify either the individuals whose points are further from the origin than the target, or are close to the target as threats.
- Classify the t closest neighbors of the target in two- or three- dimensional space as threats.
- Classify the individuals whose points fall in a cone from the origin centered at the target as threats (i.e. the cosine similarity used in latent semantic indexing [5]).

In all of these techniques, the target can be selected after the SVD has been computed – hence only one SVD is required.

Correlation information can be used in three ways:

- The selection mechanisms described above can be used in the plots based only on the points correlated with a particular target. This requires only replotting and not recomputation of the SVD. However, as we will see, there is little to be gained from this.
- Individuals who are not correlated with the target can be successively removed, and the SVD repeated on the resulting smaller dataset. This typically reduces the dataset size by 75% or more, but the contraction at each repetition becomes smaller because the remaining individuals all have fairly strong correlation with the target. All SVDs after the first have to be recomputed for each target because the winnowed datasets are target dependent.

- The sizes of the datasets remaining after each round themselves provide information. If the target does not have an unusual correlation with other individuals then the contracted datasets shrink in size quite slowly. When there is an unusual correlation of other individuals with the target, the contracted datasets tend to become smaller quite rapidly.

6 Experiments

In the experiments that follow, the part of the matrix A representing normal individuals will consist of 1000 rows and 30 columns. The 30 columns represent a set of attributes about each individual – we assume that these are intrinsic attributes and that a threat is forced to correlate with a target in the values of at least some of these attributes. Each dataset has a small number of additional rows added to represent a terrorist group. The results presented are qualitative, partly because there are too many free parameters to make an exhaustive analysis straightforward, and partly because there is not yet agreement about what structures in datasets are plausible. However, the results are for the first random dataset of each kind generated – no selection of datasets to provide better than average results was made.

In plots of two- or three-dimensional space, points corresponding to normal individuals are shown as (blue) dots, the target is shown as a (red) star, and the points corresponding to terrorist as (blue) squares.

Experiment 1. We begin with a dataset in which the points corresponding to ordinary individuals are generated distributed normally around the origin with variance 1. A terrorist group of size 10 is generated distributed normally with variance 1 around one of the normal individuals. Figure 1 shows how SVD can detect a small cluster against a background cluster. Here we assume that no target is specified beforehand – the labelling confirms the fairly clear presence of a small cluster to the left of the main cluster.

Figure 2 plots only those points that are correlated with the target. This is, of course, artificial since we are assuming that we do not know the target. However, it illustrates what selection for correlation is doing – it removes many points but does not help much with identifying the terrorist cluster because the points that are removed are far from the target in the transformed space.

Experiment 2. In the previous experiment, the terrorist cluster had the same variance as the base cluster. Hence, it is likely that points from the terrorist cluster will be overrepresented among points far from the origin. We now show that this is not the reason for

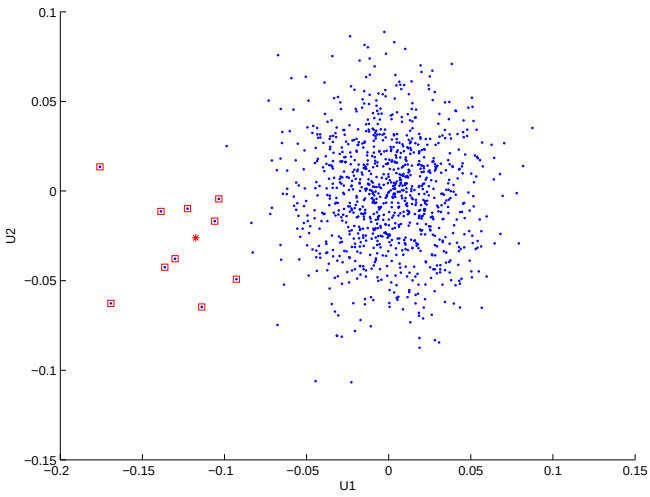

Figure 1: Plot of first two dimensions of the transformed space. Individuals normally distributed with mean 0 and variance 1, one individual randomly chosen as target, terrorists normally distributed around that individual in 30 dimensions with variance 1.

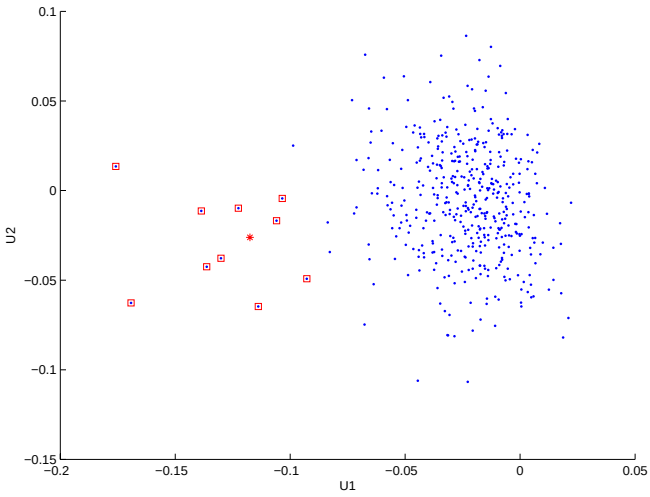

Figure 2: Same plot as in Figure 1 showing only individuals correlated with the target. Note the change of scale on the axes.

the quality of the SVD plot by repeating the experiment with the variance of the terrorist cluster at 0.5. We now expect points from the terrorist cluster to remain inside the background cluster on average.

Figure 3 shows that the presence of an outlier cluster in the transformed space is as clear as it was before. Figure 4 plots only those points correlated with the target and, as before, the separation of the target cluster is a little clearer.

In these datasets, SVD discovers the terrorist clus-

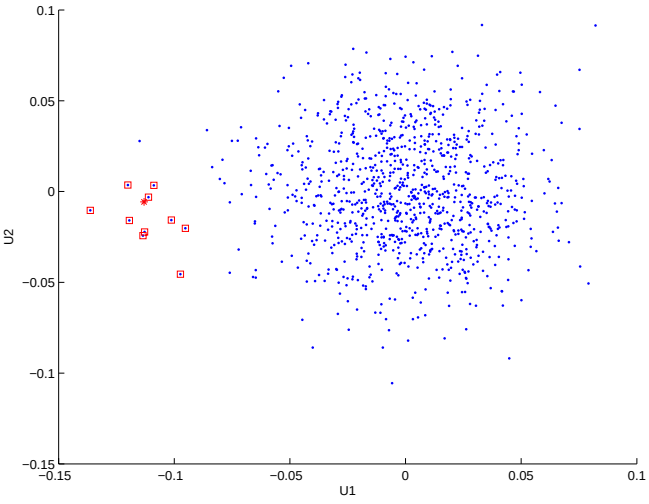

Figure 3: Individuals normally distributed with mean 0 and variance 1, one individual randomly chosen as target, terrorists normally distributed around it with variance 0.5.

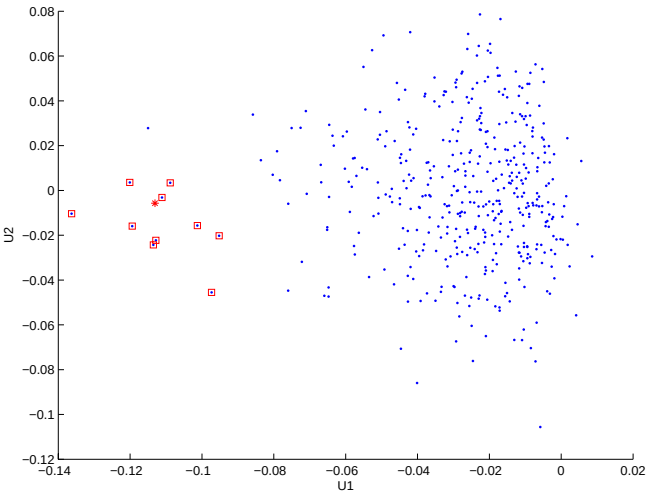

Figure 4: Same plot showing only individuals correlated with the target.

ter without knowing the target because, by Fact 1, the base cluster has little linear structure. The first singular value is overwhelmingly likely to point towards the median of the smaller embedded cluster around the target even if its points are entirely inside the larger cluster.

This is arguably an easy dataset, but not entirely trivial because the fuzzy central limit theorem suggests that, given enough data, and given that normalization takes place after the data is collected, we can expect that many parts of a dataset should look as if they were generated by a normal distribution.

Experiment 3. We now consider a dataset with following structure: 100 points are generated, normally distributed around 0 with variance 1. 100 clusters of 10 points are generated, normally distributed with variance 1 with centers at each of the original points. A terrorist cluster of size 10, normally distributed with variance 1 is generated around a random one of the second level points. So rather than a single background cluster around zero, we have a large set of background clusters with many different centers.

Figure 5 shows the resulting plot. It is clear that the terrorist cluster cannot be distinguished from the background without prior knowledge of the target, and only imperfectly then. Even when the uncorrelated points are removed from the plot (Figure 6), the terrorist cluster is not cleanly separated either by projection along a line to the target or by proximity to the target. However, the target cluster is fairly well identified using cosine similarity to the target (we have shown a cluster that would include the entire terrorist group, but many cones with smaller angles would also detect several members of the group).

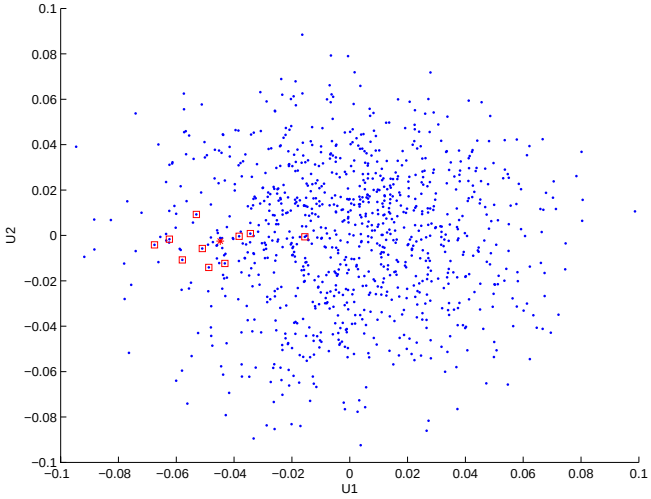

Figure 5: 10 normal clusters with variance 1 with centers drawn from a normal distribution with mean 0 and variance 1; terrorist cluster normally distributed around a randomly chosen individual with variance 1.

Experiment 4. Figure 7 shows the plot for the dataset of Experiment 3 when the row corresponding to the target is scaled by a factor of 1.2. There is very little difference between this plot and those where the target row is unweighted.

However, we now repeat the SVD using only those 422 rows of the original matrix that are correlated with the target. The results are shown in Figure 8. Both projection onto a vector from the origin to the

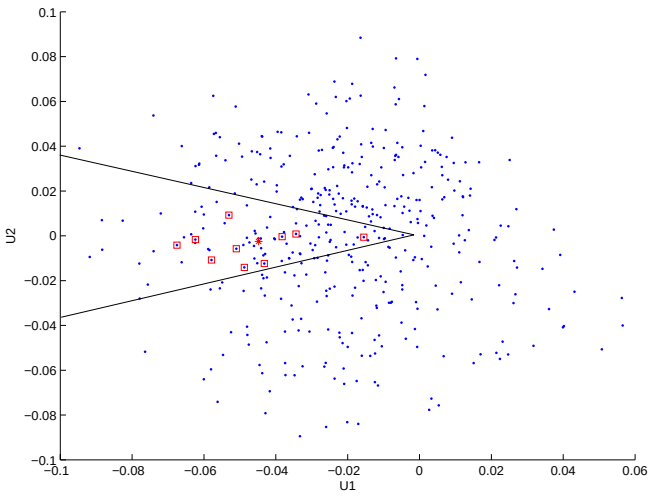

Figure 6: Same plot showing low false positive rate using a cone aimed at the target as a selector.

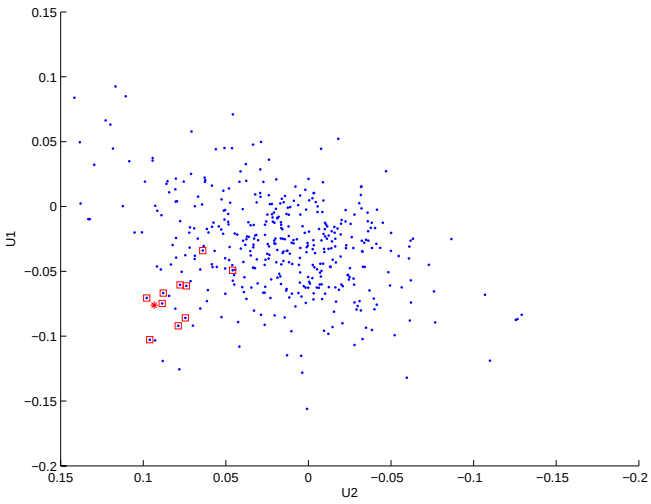

Figure 8: Repeated SVD using only the 422 individuals correlated with the target in the first round.

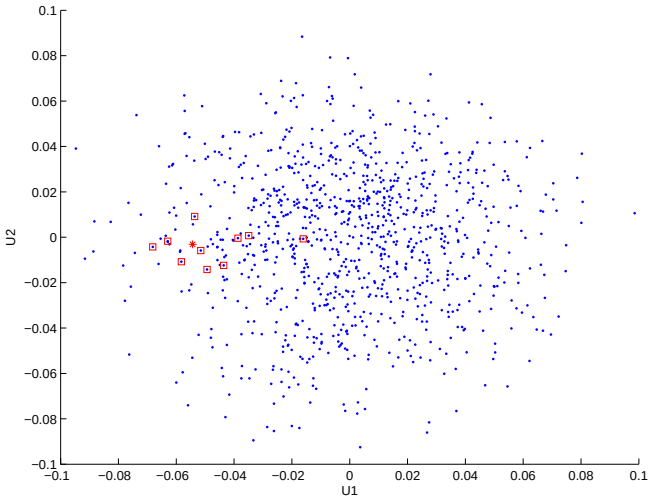

Figure 7: Same dataset as for Experiment 3, with the target row scaled by 1.2.

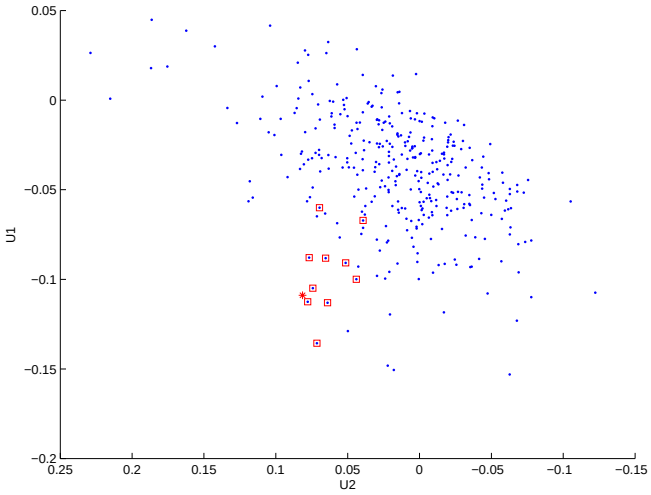

Figure 9: Repeated SVD using only the 362 individuals correlated with the target.

target, and proximity now begin to discover the terrorist cluster.

Figure 9 shows what happens after a third round of SVD on the 362 points correlated with the target on the previous round. Both projection onto a vector and proximity continue to improve their predictive performance, and now clearly identify the terrorist group. Notice the flattening of the number of uncorrelated points being removed at each stage.

Experiment 5. Using the dataset of Experiment 3, we now multiply the row corresponding to the target by 4. Figure 10 shows what happens – the target point moves far from the origin, but it also tends to pull the correlated points towards it, and so away from the main cluster. Both proximity and proximity on the projection onto the vector from origin to target are effective at finding the terrorist cluster.

Experiment 6. In the previous experiments (Experiments 3–5), the terrorist cluster was still distinguished because it was the only cluster at the 'third' level. We now generate a dataset with 100 points, normally distributed around 0 with variance 1. 100 clusters of 10 points are generated, normally distributed with variance 1 around each of the original points, and 20 clusters of size 10 normally distributed with variance 1 are generated around randomly chosen points in the

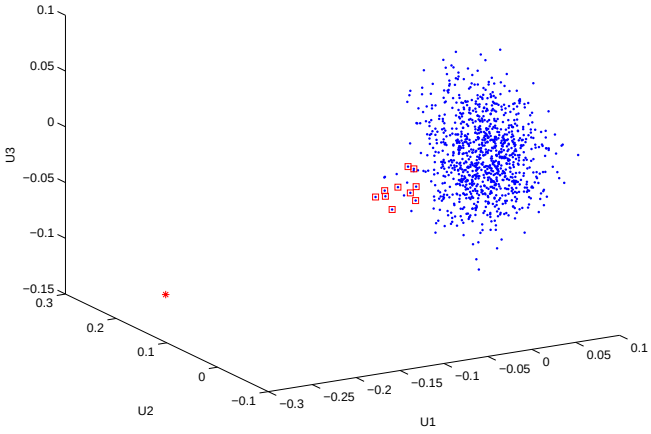

Figure 10: 10 normal clusters with variance 1 with centers drawn from a normal distribution with mean 0 and variance 1; terrorist cluster normally distributed around a randomly chosen individual with variance 1; weight of 4 on the target.

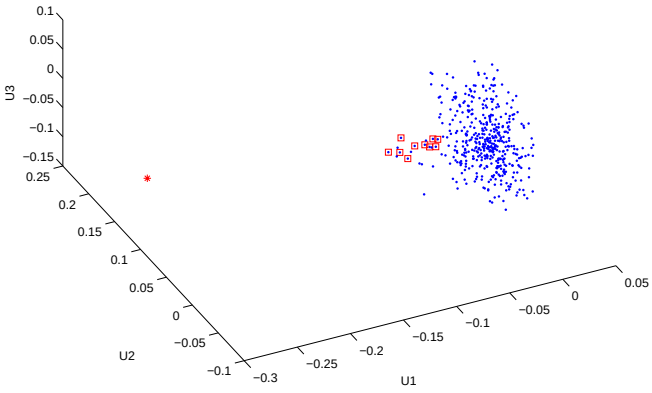

Figure 11: Same plot showing only the individuals correlated with the target.

second level. One of these 'third' level clusters is chosen as the terrorist cluster and its center as the target.

Figure 12 shows that both projection and proximity find the terrorist cluster with reasonable accuracy.

Experiment 7. In the dataset, the local environment of each of the second level cluster centers is the same and we can choose any of them as possible terrorist clusters. On the other hand, the local environment of all of the other points is quite different. Figure 13 shows the sizes of the sets of points correlated with a particular point, when that point is a second-level cluster center (a possible target) and when it is one of the other points.

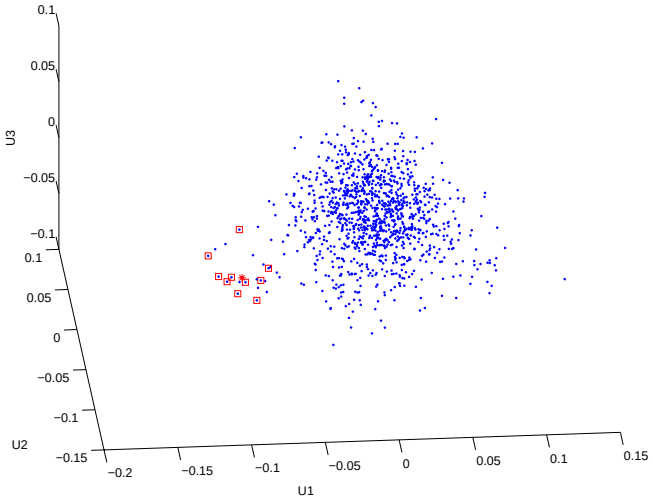

Figure 12: Three levels of clusters: first level of 100 cluster centers normally distributed around 0 with variance 1; second level of 10 cluster centers normally distributed with variance 1 around these; then 20 clusters of size 10 distributed around these. One second-level cluster center designated as the target.

Those points that are potential targets have neighborhoods that start out smaller and shrink more rapidly than the neighborhoods of points that are not targets. The difference between the two types of points is marked, even by the third round.

After	Size of sets correlated with a point						
rnd	that is a target			that is not a target			
1	145	419	199	831	370	586	416
2	20	27	47	513	90	194	150
3			20	461	48	86	78
4				400	42	56	65

Figure 13: Sizes of correlated sets after elimination of uncorrelated individuals. Initial size of all sets is 1200.

Experiment 8. In our experiments so far, the number of terrorists has been about 1% of the total number of individuals. This fraction is too large to be realistic, even if a substantial prescreening process is applied before this kind of data mining is used.

Figure 14 shows the three-dimensional plot of a dataset with 10000 rows, normally distributed around the origin with variance 1, with a 10-terrorist cluster normally distributed with variance 1 generated around one of the ordinary individuals. The terrorist cluster is now much more diffuse. However, note that the extremal point along the projection on the vector from the origin to the target is a terrorist, and several others project on this vector further from the origin than the target.

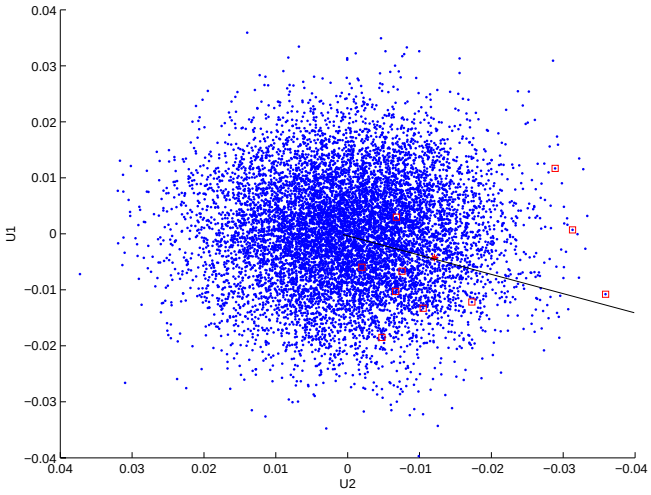

Figure 14: Plot of a 10-terrorist group and a dataset of 10000 ordinary individuals.

As Figure 15 shows, adding a weight of 4 to the target clearly selects the majority of the terrorist group, so even in large datasets SVD is effective.

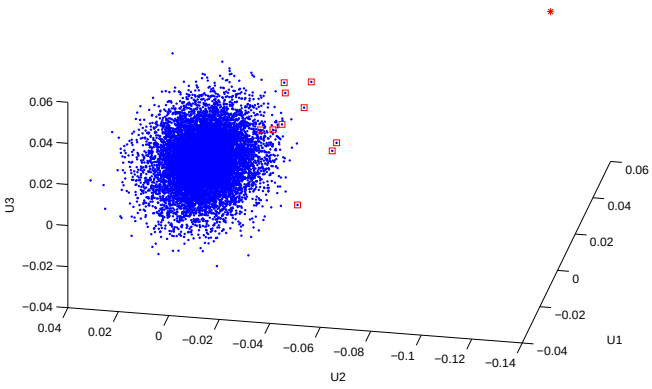

Figure 15: Same dataset with the target weight scaled by 4.

Experiment 9. Fact 1 suggests that sparseness in datasets will not cause difficulties for SVD. This illustrates one of the strong properties of SVD – it is capable of detecting correlation even between individuals who have no (non-zero values of) attributes in common, via higher-order correlations.

Figure 16 shows a plot of a dataset similar to that of Experiment 1, but with 80% of the values set to zero. Although many of the terrorist cluster are not close to the target, several members still are. A dataset like this represents a situation where underlying attributes exist but are missing for some reason.

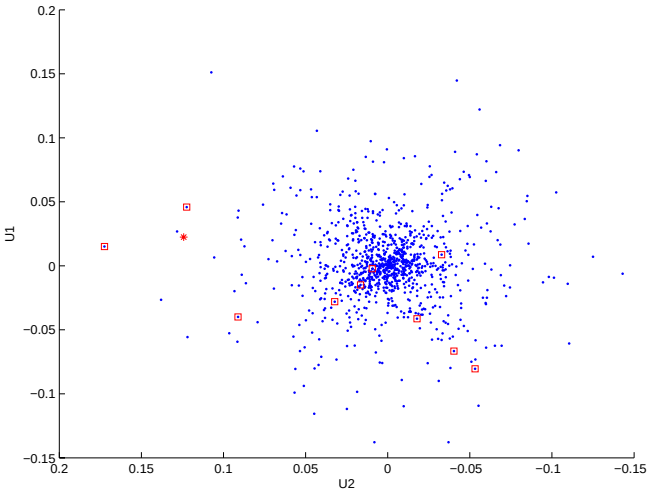

Figure 16: A sparse dataset generated by setting 80% of the values in a dense dataset from Experiment 1 to zero.

Another kind of sparse dataset is one in which there are no meaningful values for the zero entries. Figure 17 shows the plot of such a dataset. Here all 1010 rows are generated using a normal distribution with mean zero and variance 1, and a random row among the first 1000 is selected as the target. The rows of the terrorist cluster are then correlated with the target in the following way: if a target attribute has a non-zero value then, with 70% probability, the terrorist row is changed to a value drawn from a normal distribution whose mean is the value of the target attribute and whose variance is 1; otherwise the value is left unchanged. The correlation of the terrorist cluster with the target is plainly visible.

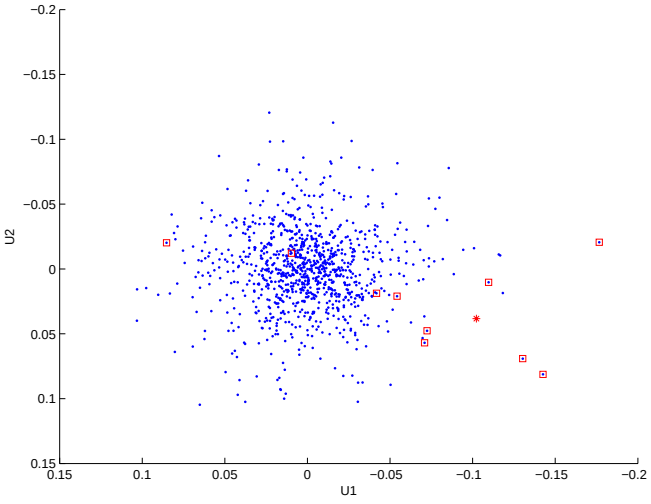

Figure 17: A sparse dataset generated directly.

Experiment 10. We now show that similar effects hold for distributions other than the normal distribution. The Poisson distribution with mean 1 generates many values close to 1, with the frequency decreasing rapidly with magnitude. We build a dataset of a 1000 rows from this distribution, subtracting λ to make the values approximately zero mean.

Figure 18 shows the results when the terrorist cluster is generated using a normal distribution with variance 1 around a randomly chosen row. Figure 19 shows the results when the terrorist cluster is also generated by the same Poisson distribution around the target (i.e the mean of the terrorist distribution is roughly the target).

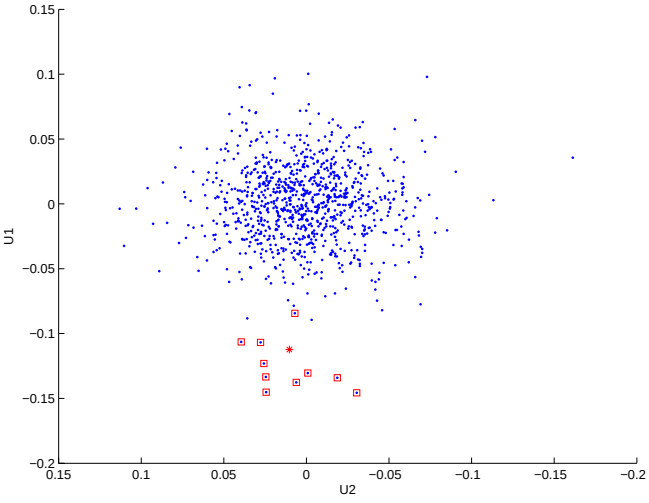

Figure 18: Ordinary individuals generated from a Poisson distribution, terrorist cluster normally distributed around a randomly chosen target.

7 Related Work

Techniques used for detecting outlying objects or outlying processes, for example Independent Component Analysis (ICA) [8], and 1-Class Classification [15, 16] seem less likely to provide good solutions for terrorism detection, although they may be effective when the *values* are completely beyond the control of individuals.

Singular value decomposition has been known since 1873, and used extensively in computing since Golub discovered an effective algorithm to compute it [7]. There is a vast literature on its use for dimensionality reduction; and it has been used for information retrieval where proximity (in the sense of cosine similarity) serves as a proxy for correlation [3].

Social Network Analysis [6] study the interactions between individuals and derives global properties from the structure of the resulting graphs. There are two

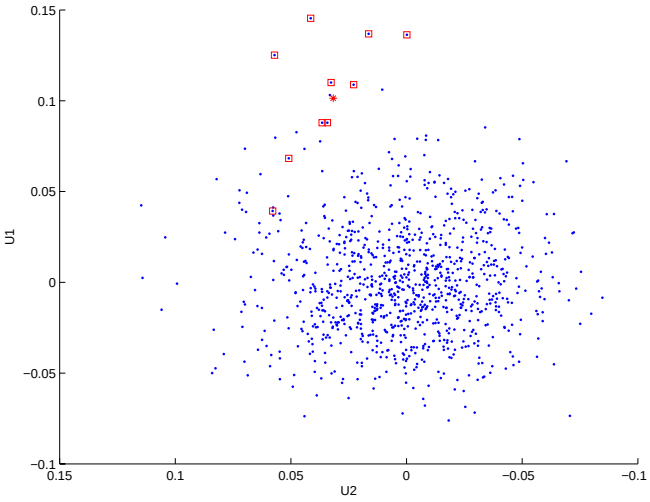

Figure 19: Ordinary individuals generated from a Poisson distribution, terrorist cluster Poisson distributed around a randomly chosen target.

serious drawbacks to the use of SNA techniques for terrorism detection:

- Networks are built by adding pairwise links between two individuals, and this will not scale well since the number of potential links is quadratic in the number of individuals. In practice, SNA seems to have been used when a particular individual threat has been identified, as a tool to discover his or her collaborators. In other words, SNA has a bootstrap problem (but may be useful once the kind of prescreening we suggest here has been applied).

- Links are made between individuals as the result of some interaction between them, rather than because of some correlation between them. In other words, SNA may discover two collaborators who *meet* at a target site, but will not discover them simply because they both *visit* the target site.

Link or traffic analysis has similar drawbacks: it can be useful once at least one member of a terrorist group has been identified; but it has the same limitation of only detecting direct relationships between two individuals, rather than their correlated actions. Traffic analysis has been used to detect unusually strong patterns of interaction, but only on the basis of a handful of attributes.

The paper [13] describes experiments using Inductive Logic Programming on relational datasets recording nuclear smuggling and contract killing. This work could presumably be generalized to counterterrorism.

8 Conclusion

We have shown that SVD is able to detect small correlated clusters, representing terrorists, against a variety of backgrounds representing degrees of innocent correlation. Qualitatively, in every case there exists a mechanism that identifies at least one (usually more) of the terrorist cluster based on proximity to the target, either directly in a low-dimensional space or by projection along a vector derived from the target. The number of false positives induced by these procedures is not trivial, but it is arguably reasonable. Our results do not suggest an optimal strategy for applying SVD for terrorist detection – rather they suggest a number of effective techniques. More experience will be required to determine how best to combine these techniques.

Many questions remain: are the generated datasets used for these experiments reasonable analogues of real-world datasets, can the *ad hoc* detection procedures used here be codified and automated, and does the performance remain acceptable as datasets become larger, perhaps much larger?

ACKNOWLEDGEMENT. This work was carried out while the author was a guest of the Faculty of Information Technology, University of Technology, Sydney.

References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2001.

[2] Y. Azar, A. Fiat, A.R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *ACM Symposium on Theory of Computing*, pages 619–626, 2001.

[3] M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[4] S. Chakrabarti and A. Strauss. Carnival booth: An algorithm for defeating the computer-assisted passenger screening system. Course Paper, MIT 6.806: Law and Ethics on the Electronic Frontier, `http://www.swiss.ai.mit.edu/6805/student-papers/spring02-papers/caps.htm`, 2002.

[5] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[6] L. Garton, C. Haythornthwaite, and B. Wellman. Studying online social networks. *Journal of Computer-Mediated Communication*, 3(1), 1997.

[7] G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[8] A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

[9] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. In *Proceedings of the 41st Foundations of Computer Science (FOCS '00)*, page 367, 2000.

[10] A. Kontostathis and W.M. Pottenger. Detecting patterns in the LSI term-term matrix. Technical Report LU-CSE-02-010, Department of Computer Science and Engineering, Lehigh University, 2002.

[11] A. Kontostathis and W.M. Pottenger. Improving retrieval performance with positive and negative equivalence classes of terms. Technical Report LU-CSE-02-009, Department of Computer Science and Engineering, Lehigh University, 2002.

[12] V.E. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[13] R.J. Mooney, P. Melville, L.R. Tang, J. Shavlik, I de Castro Dutra, D. Page, and V.S. Costa. Relational data mining with Inductive Logic Programming for link discovery. In *Proceedings of the National Science Foundation Workshop on Next Generation Data Mining*, November 2002.

[14] M. Newman and D. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263:341–346, 1999.

[15] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, Microsoft Research, 1999.

[16] D.M.J. Tax. *One Class Classification*. PhD thesis, Technical University Delft, 2000.

Homeland Defense, Privacy-Sensitive Data Mining, and Random Value Distortion

Souptik Datta* Hillol Kargupta† Krishnamoorthy Sivakumar‡

Abstract

Data mining is playing an increasingly important role in sifting through large amount of data for homeland defense applications. However, we must pay attention to the privacy issues while mining the data. This has resulted in the development of several privacy-preserving data mining techniques. The random value distortion technique is one among them. It attempts to hide the sensitive data by randomly modifying the values. This paper questions the utility of the random value distortion technique. The paper develops a random matrix-based spectral filtering technique to retrieve original data from the dataset distorted by adding random values. The proposed method works by comparing the spectrum generated from the observed data with that of random matrices. The paper presents the theoretical foundation and extensive experimental results to demonstrate that the random value distortion technique may not preserve any data privacy after all.

Keywords: Privacy preserving data mining, value perturbation, random matrices, eigenanalysis.

1 Introduction

Many homeland defense applications require mining heterogeneous data for creating profiles, constructing social network models, detecting terrorist communications among others. Usually the data is very sensitive to privacy issues. Financial transactions, healthcare records, and network communication traffic are a few examples. Data mining in such privacy-sensitive domains is facing growing concerns. Therefore, we need to develop data mining techniques that are sensitive to the privacy issue. This has fostered the development of a class of data mining algorithms [1, 9] that try to protect the data privacy with varying degrees of success. Most of these algorithms try to extract the data patterns without directly accessing the original data and guarantees that the mining process does not get sufficient information to reconstruct the original data.

This paper explores the random value perturbation-based approach [1], a well-known technique for masking the data using random noise. The idea is to preserve data privacy by adding random noise, while making sure that the random noise still preserves the signal from the data so that the patterns can be closely estimated. This paper questions the privacy-preserving capability of the random value perturbation-based approach and shows that the original data can be accurately estimated from the perturbed data using a spectral filter that exploits some theoretical properties of random matrices. It presents the theoretical foundation and provides experimental results to support this claim.

Section 2 offers an overview of the related literature in privacy preserving data mining. Section 3 describes the random data perturbation method proposed in [1]. Section 4 discusses the theoretical foundation of our approach that relies on known properties of random matrices. Section 5 describes the random matrix-based eigen analysis methods to extract the original dataset. Section 6 applies the proposed technique and reports its performance for various data sets. Finally, Section 7 concludes this paper and outlines future research directions.

2 Related Work

There exists a growing body of literature on privacy-sensitive data mining. These algorithms can be divided into two different groups. One approach adopts a distributed framework; the other approach adds random noise to the data in such a way that the individual data values are distorted while still preserving the underlying distribution properties at a macroscopic level. The following part of this sections briefly discusses these two approaches.

The distributed approach supports computation of data mining models and extraction of "patterns" at a given node by exchanging only the minimal necessary information

*Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, USA.

†Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, USA.

‡School of Electrical Engineering and Computer Science, Washington State University, USA.

among the participating nodes without transmitting the raw data. The field of distributed data mining [12, 19] produced several distributed algorithms that are sensitive to privacy. For example the meta-learning based JAM system [25] was designed for mining multi-party distributed sensitive data such as financial fraud detection. The Fourier spectrum-based approach to represent and construct decision trees [13, 18], the Collective hierarchical clustering [7] are examples of additional distributed data mining algorithms that can be used with minor modifications for privacy-preserving mining from distributed data. In the recent past, several distributed techniques to mine multi-party data have been reported. A privacy preserving technique to construct decision trees [21] proposed elsewhere [15], multi-party secured computation framework [20], association rule mining from homogeneous [9] and heterogeneous [27] distributed data sets are some examples. [11] reports privacy preserving correlation computaion between datasets owned by multiple parties. There also exists a collection of useful privacy-sensitive data mining primitives such as secure sum computation [22], secure scalar product computation [27].

There is also a somewhat different approach and the algorithms belonging to this group works by first perturbing the data using randomized techniques. The perturbed data is then used to extracts the patterns and models. The randomized value distortion technique for learning decision trees [1] and association rule learning [3] are examples of this approach. Additional work on randomized masking of data can be found elsewhere [26].

This paper explores the second approach [1] that works by adding random noise to the data set in order to hide the individual data values of different attributes. It points out that in many cases the noise can be separated from the perturbed data by studying the spectral properties of the data and as a result its privacy can be seriously compromised. Before presenting the technique to do that, let us review the randomized value distortion [1] technique in details.

3 Random Value Perturbation Technique: A Brief Review

For the sake of completeness, we now briefly review the random data perturbation method suggested in [1]. We also discuss the procedure for reconstructing the original data distribution, as suggested in [1].

3.1 Perturbing the Data

The random value perturbation method attempts to preserve privacy of the data by modifying values of the sensitive attributes using a randomized process [1]. The authors explore two possible approaches - Value-Class Membership and Value Distortion - and emphasize the Value Distortion approach. In this approach, the owner of a dataset returns a value $x_i + r$, where x_i is the original data, and r is a random value drawn from a certain distribution. Most commonly used distributions are the uniform distribution over an interval $[-\alpha, \alpha]$ and Gaussian distribution with mean $\mu = 0$ and standard deviation σ. The n original data values $x_1, x_2, \ldots, x_n$ are viewed as realizations of n independent and identically distributed (i.i.d.) random variables X_i, $i = 1, 2, \ldots, n$, each with the same distribution as that of a random variable X. In order to perturb the data, n independent samples $r_1, r_2, \ldots, r_n$, are drawn from a distribution R. The owner of the data provides the perturbed values $x_1 + r_1, x_2 + r_2, \ldots, x_n + r_n$ and the cumulative distribution function $F_R(r)$ of R. The reconstruction problem is to estimate the distribution $F_X(x)$ of the original data, from the perturbed data.

3.2 Estimation of Distribution Function from the Perturbed Dataset

The authors [1] suggest the following method to estimate the distribution $F_X(x)$ of X, given n independent samples $w_i = x_i + r_i$, $i = 1, 2, \ldots, n$ and $F_R(r)$. Using Bayes' rule, the posterior distribution function $F'_X(x)$ of X, given that $X + R = w$, can be written as

$$F'_X(x) = \frac{\int_{-\infty}^{x} f_Y(w - z) f_X(z) dz}{\int_{-\infty}^{\infty} f_Y(w - z) f_X(z) dz},$$

which upon differentiation with respect to x yields the density function

$$f'_X(x) = \frac{f_Y(w - x) f_X(x)}{\int_{-\infty}^{\infty} f_Y(w - z) f_X(z) dz}.$$

If we have n independent samples $x_i + r_i = w_i$, $i = 1, 2, \ldots, n$, the corresponding posterior distribution can be obtained by averaging:

$$f'_X(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{f_Y(w_i - x) f_X(x)}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X(z) dz}. \qquad (1)$$

For sufficiently large number of samples n, we expect the above density function to be close to the real density function $f_X(x)$. In practice, since the true density $f_X(x)$ is unknown, we need to modify the right-hand side of equation (1). The authors suggest an iterative procedure where at each step $j = 1, 2, \ldots$, the posterior density $f_X^{j-1}(x)$ estimated at step $j-1$ is used in the right-hand side of equation (1). The uniform density is used to initialize the iterations. The iterations are carried out until the difference between successive estimates becomes small. In order to speed up computations, the authors also discuss approximations to the above procedure using partitioning of the domain of data values.

4 Theory of Random Matrices

In this section, we discuss the general theory of random matrices that is used to filter the noise from the perturbed dataset to obtain an estimate of the actual dataset. Our filtering approach is based on the observation that the distribution of eigenvalues of random matrices [17] exhibit some well known characteristics.

A random matrix is a matrix whose elements are random variables with given probability laws. The theory of random matrices deals with the statistical properties of the eigenvalues of such matrices. Eigenvalues of random matrices offer many interesting properties. For example, Wigner's semicircle law, which says if X is an $n \times n$ matrix and has i.i.d. entries with zero mean and unit variance, the distribution of eigenvalues of $\frac{X+X'}{2\sqrt{2n}}$ has a probability density function given by

$$f(x) = \begin{cases} \frac{1}{\pi}(2n - x^2)^{1/2}, & |x| < \sqrt{2n} \\ 0, & \text{otherwise.} \end{cases}$$

In this paper, we are mainly concerned about distribution of eigenvalues of the sample covariance matrix obtained from a random matrix. Let X be a random $m \times n$ matrix whose entries are X_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, are i.i.d. random variables with zero mean and variance σ^2. The covariance matrix of X is given by $Y = \frac{1}{m}X'X$. Clearly, Y is an $n \times n$ matrix. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of Y. Let

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} U(x - \lambda_i),$$

be the empirical cumulative distribution function (c.d.f.) of the eigenvalues λ_i, $(1 \leq i \leq n)$, where

$$U(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

is the unit step function. In order to consider the asymptotic properties of the c.d.f. $F_n(x)$, we will consider the dimensions $m = m(N)$ and $n = n(N)$ of matrix X to be functions of a variable N. We will consider asymptotics such that in the limit as $N \to \infty$, we have $m(N) \to \infty$, $n(N) \to \infty$, and $\frac{m(N)}{n(N)} \to Q$, where $Q \geq 1$. Under these assumptions, it can be shown that [8] the empirical c.d.f. $F_n(x)$ converges in probability to a continuous distribution function $F_Q(x)$ for every x, whose probability density function (p.d.f.) is given by

$$f_Q(x) = \begin{cases} \frac{Q\sqrt{(x-\lambda_{\min})(\lambda_{\max}-x)}}{2\pi\sigma^2 x} & \lambda_{\min} < x < \lambda_{\max} \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

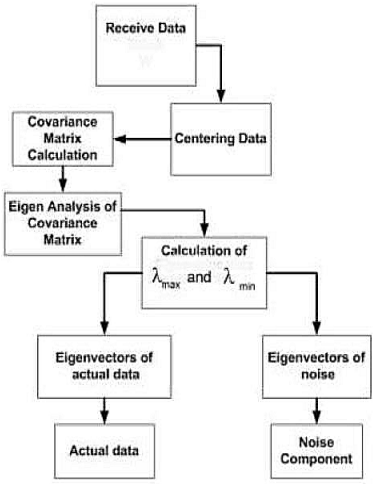

Figure 1. Main steps of the proposed spectral filtering technique.

where $\lambda_{\min} = \sigma^2(1 - 1/\sqrt{Q})^2$ and $\lambda_{\max} = \sigma^2(1 + 1/\sqrt{Q})^2$. Further refinements of this result and other discussions can be found in [24, 5, 16, 2, 4, 14, 28, 23].

5 Random Matrix-Based Data Filtering

Suppose actual data S is perturbed by a noise random variable R to produce $W = S + R$. Let $w_i = s_i + r_i$, $i = 1, 2, \ldots, m$, be m (perturbed) data points, each being a vector of n features. Thus the perturbed dataset, can be considered to be an $m \times n$ random matrix W, having n features and m instances. Our proposed filtering technique first calculates the covariance matrix of the perturbed data W. Using the distribution of eigenvalues of the covariance matrix, and the theory of random matrices, the covariance matrix of W is decomposed into a noise part and an actual data part. The eigenvectors corresponding to actual data are then used to reconstruct the actual data.

In the following section, we discuss some details of the filtering procedure. We first assume that the entire distribution $F_R(r)$ of the random noise R is known. Later, we discuss how the noise variance can be estimated from the eigenvalue distribution of the perturbed data. Details of this method and the supporting theory can be found in [10].

5.1 Known Noise Variance

When the noise distribution $F_R(r)$ of R is completely known, the noise variance σ^2 is first calculated. Equation

(2) is then used to calculate λ_{max} and λ_{min}. They provide the theoretical bounds of the eigenstates corresponding to noise. From the perturbed data, we compute the eigenvalues of its covariance matrix Y, say $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then we identify the noisy eigenstates $\lambda_i \leq \lambda_{i+1} \leq \cdots \leq \lambda_j$ such that $\lambda_i \geq \lambda_{min}$ and $\lambda_j \leq \lambda_{max}$. The remaining eigenstates are the eigenstates corresponding to actual data. Let, V_r = diag $(\lambda_i, \lambda_{i+1}, \ldots, \lambda_j)$ be the diagonal matrix with all noisy eigenvalues, and A_r be the matrix whose columns are eigenvectors corresponding to the eigenvalues in V_r. Similarly, let V_s be the eigenvalue matrix for the actual data part and A_s be the corresponding eigenvector matrix which is a $n \times k$ matrix ($k \leq n$). Based on these matrices, we decompose the covariance matrix Y into two parts, Y_s and Y_r with $Y = Y_s + Y_r$, where $Y_r = A_r V_r A_r'$, is the covariance matrix corresponding to random noise part, and $Y_s = A_s V_s A_s'$, is the covariance matrix corresponding to actual data part. An estimate $\hat{S}$ of the actual data S is obtained by projecting the data W on to the subspace spanned by the columns of A_s. In other words, $\hat{S} = W A_s A_s'$. Figure 1 shows the flow digram of the process described above.

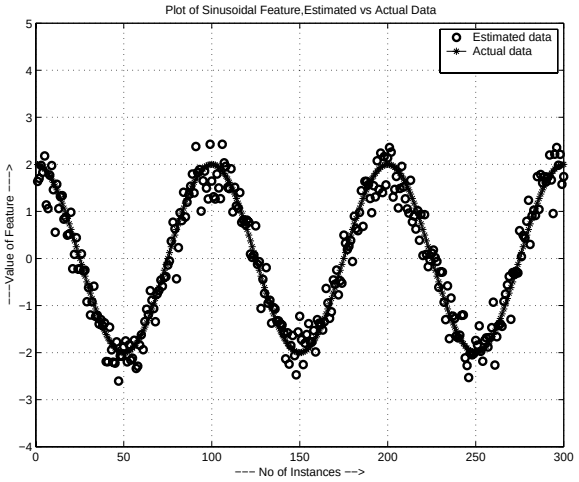

Figure 2. Estimation of original sinusoidal data with known random noise variance.

5.2 Unknown Noise Variance

When the noise variance σ^2 is unknown, we first estimate it using the perturbed data. The estimated noise variance is then used to filter the perturbed data. In order to estimate the noise variance σ^2 we first compute the eigenvalues of the covariance matrix Y of the perturbed data W. A histogram of the eigenvalue distribution is plotted and compared to that of the theoretical noise eigenvalue density function $f_Q(x)$ given in equation (2). Note that the density function $f_Q(x)$ depends on the variance σ^2. Typically,

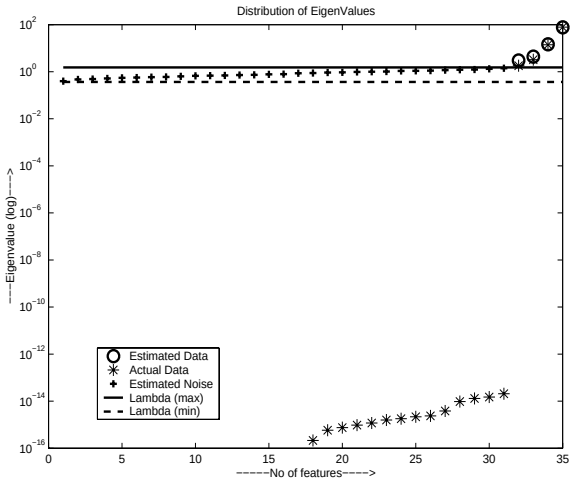

Figure 3. Distribution of eigenvalues of actual data , and estimated eigenvalues of random noise and actual data.

the theoretical density function $f_Q(x)$ is a good fit to the left portion of the histogram of the computed eigenvalues, corresponding to small eigenvalues. The larger eigenvalues that do not fit this theoretical density function correspond to the actual information part of the perturbed data. An iterative procedure is employed to obtain the value of σ that results in the best fit of $f_Q(x)$ to the observed histogram.

6 Experimental Results

Our proposed method is used on datasets of different sizes which have some trend in their values. The actual dataset is distorted by adding Gaussian noise (Normally distributed random numbers with zero mean and specific variance), and our proposed technique is applied to recover the actual data from the perturbed data with known noise variance. Experimental results show that the proposed method estimates the individual data closely and also detects the pattern of actual data. Figure 2 shows one such estimation of data when the actual data has sinusoidal trend.

The distribution of eigenvalues shows (Figure 3) the method accurately distinguishes between noisy eigen values and eigenvalues corresponding to actual data. Note that the estimated eigenvalues of actual data is very close to eigenvalues of actual data and almost overlap with them above λ_{max}. Though there are no estimates of the eigenvalues of actual data below the λ_{min} axis, the estimation of actual data is fairly accurate, because the magnitudes of those eigenvalues are very small and hence do not have any significant impact on the whole estimation.

We used a dataset of 300 instances and 20 features which has definite trend in its features. We added a Gaussian ran-

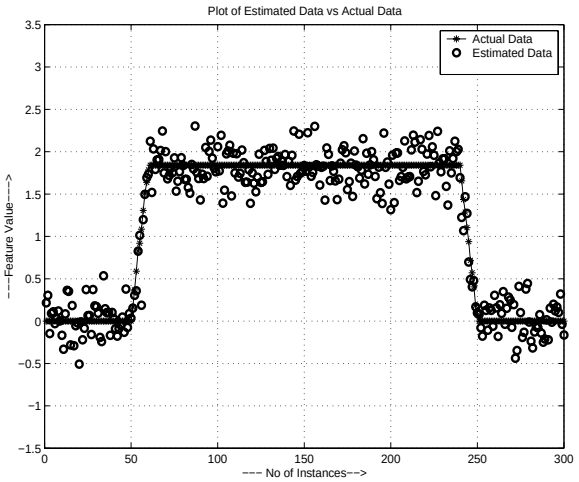

Figure 4. Estimated dataset preserves the 'Plateau' trend of original data.

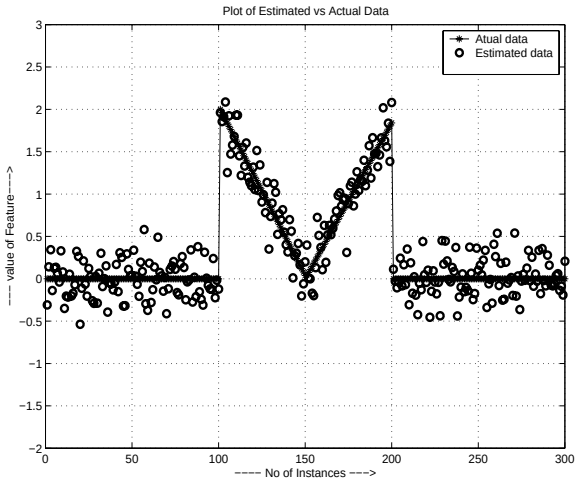

Figure 5. Estimated dataset preserves the 'Triangular' trend of original data.

dom noise with mean 0 and standard deviation $\sigma = 0.25$ to each of these data values and applied our algorithm to recover the actual data from the distorted data with known noise $\sigma = 0.25$. Figure 4, Figure 5 show estimations of dataset with different types of trends in their actual values. The actual datasets have trends like plateau and triangles. The proposed method estimates the values closely and preserve the original trend of the actual dataset.

Quality of recovery depends upon relative noise content of the data. The method performs poorly if relative noise-content in the actual data is too high. We define the term 'Signal-to-Noise Ratio' (SNR) to quantify the relative amount of noise added to actual data to perturb it.

$$\text{SNR} = \frac{\text{Value of Actual Data}}{\text{Value of Noise Added to the Data}}$$

As the noise added to the actual value increases, the SNR decreases. Our experiments show that this method predicts the actual data reasonably well up to a SNR value of 1.0 (i.e. 100% noise). The results shown in figures 2, 4, 5 are the case of mean SNR value nearly 2, i.e. 50% noise. As the SNR goes below 1, the estimation becomes too erroneous. Figure 6 shows the difference in estimation as the SNR increases from 1. The upper figure shows the estimation corresponding to 100% noise (mean SNR = 1), and the lower figure shows estimation corresponding to 16.67% noise (mean SNR = 6).

In case of unknown noise distribution, the method estimates the noise variance first. From the eigenvalues of covariance matrix of actual data, a histogram of the eigenvalue distribution is obtained, and this is compared with best possible theoretical density function given by Equation 2. The variance corresponding to the best fit gives the estimation of the noise variance.

To get the best estimation of variance, the algorithm estimates noise variance from the best fit curve several times. In each trial , the variance estimation algorithm starts with a very small variance value near zero, create the theoretically generated distribution and measures the mean square error between it and histogram of eigenvalues of actual data. It then increases variance by a small value, again computes the mean square error and compares it with the previous error to get the minimum error and corresponding variance. The algorithm does the said operation up-to a threshold value of variance, and stores of the variance corresponding to minimum mean square error between theoretically generated density function curve and histogram of eigenvalues of actual data.That value of variance is treated as the estimated value of noise variance for that particular trial.In our experiment, we used 100 such trials for each variance estimation. After the set of estimates are calculated from all trials, the distribution of estimated variances is checked for outliers in them. The mean μ_1 and standard deviation σ_1 of the estimates are calculated , and values lying outside the span $\mu_1 \pm 2\sigma_1$ are discarded. During each trial, if the algorithm does not get best fit within a predefined threshold value of variance, it stores that threshold value of variance as the estimation. These values are also treated as outliers at the end and are discarded.

After discarding the outlier estimations, an average of the rest of the estimates are taken to get the actual estimate of noise variance. Discarding the outliers and taking average of the remaining number of estimate improves the estimation accuracy to a large extent. Figure 7 shows the theoretical density curve and distribution of actual eigen-

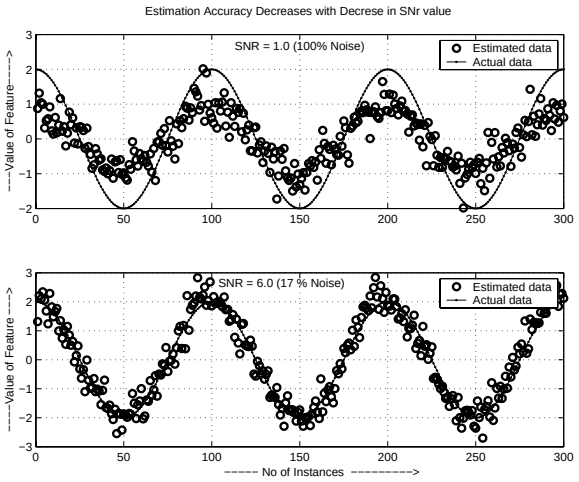

Figure 6. A higher noise content (low SNR) leads to less accurate estimation. SNR in upper figure is 1, while that for lower figure is 6.

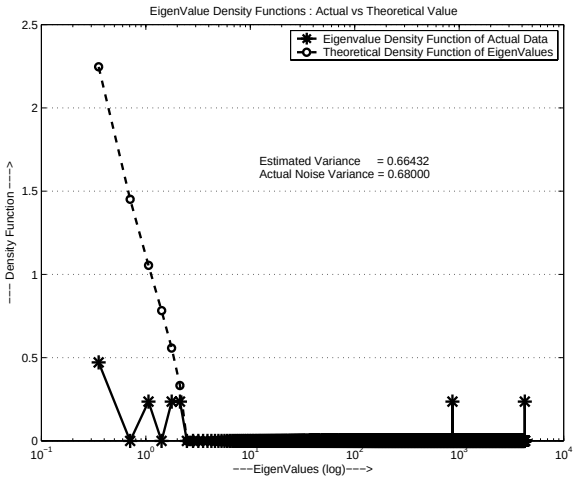

Figure 7. Theoretical density function and the actual distribution of eigenvalues.

values.The average over 100 estimates gives an estimated variance of 0.66432 where the actual noise variance is 0.68. Although not all the estimates are always so close, on an average, the difference between the estimated variance and true variance remains within 10% of the actual variance in all our experiments. Once the noise variance is estimated, the same technique is applied as before to estimate the original data. Figure 8 shows the estimation of actual data of a relatively small dataset with high SNR when distribution of noise is not known. Figure 9 displays the distribution of eigen values.

7 Conclusion and Future Work

Preserving privacy in data mining activities is a very important issue. This paper illustrates a noise filtering technique by which true data values can be estimated from the perturbed values (by random noise). This raises questions against the claim of preserving privacy by perturbing data with random numbers and disclosing the perturbed dataset as well as the probability distribution of the random number generator. The proposed approach works by comparing the empirically observed eigenvalue distribution of the given data with that of the known distribution of random matrices. The theoretically known values of upper and lower limits of the spectrum (eigenvalues) are used to identify the boundary between the eigen-states due to noise and that of the actual data.

This random matrix based approach to separate the actual data and noisy eigen-states has potential computational advantages. Indeed, since the upper bound $\lambda_{\max}$ of the noisy eigenvalues is known a priori, one can easily use a suitable numerical technique (e.g., power method [6]) to compute just the few largest eigenvalues. Once these eigenvalues and corresponding eigenvectors are computed, one can obtain the actual-data-part of the covariance matrix, which can be subtracted off from the total covariance to isolate the noise-part of the covariance. The proposed approach is simple, and retrieves actual data with reasonable precision. For the datasets considered in this paper, our experimental results support this claim. So, the method of perturbing data with random number to hide their original value is not a very reliable method to preserve privacy.

This work points out a potential problem in the existing literature. However, it leaves open the problem of coming up with methods which can actually preserve privacy without destroying statistical properties of the original dataset. We believe that this can be done by first narrowing down the specific pattern that we want to preserve through randomized perturbation. We hope that this work will encourage data mining researchers to design privacy-preserving techniques that pay careful attention to the properties of random noise and their effect on preserving privacy.

Acknowledgments

The authors acknowledge supports from the United States National Science Foundation CAREER award IIS-0093353, NASA (NRA) NAS2-37143, and TEDCO, Maryland Technology Development Center.

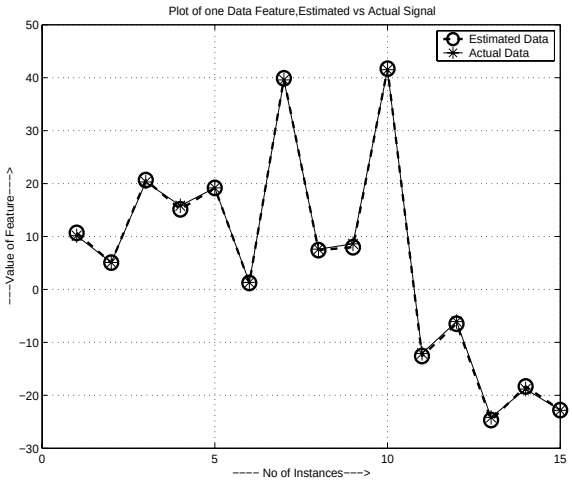

Figure 8. Estimation of actual data when the noise distribution is not known.

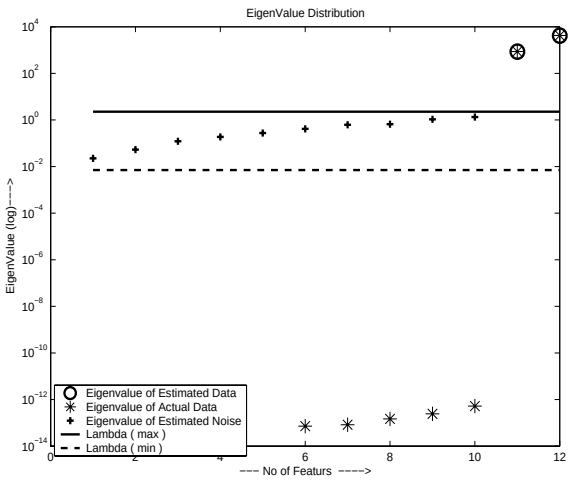

Figure 9. Distribution of eigenvalues for the estimations without the knowledge of noise distribution.

References

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceeding of the ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, Texas, May 2000. ACM Press.

[2] Z. D. Bai, J. W. Silverstein, and Y. Q. Yin. A note on the largest eigenvalue of a large dimensional sample covariance matrix. *Journal of Multivariate Analysis*, 26(2):166–168, August 1988.

[3] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery an d Data Mining (KDD)*, July 2002.

[4] S. Geman. A limit theorem for the norm of random matrices. *The Annals of Probability*, 8(2):252–261, April 1980.

[5] U. Grenander and J. W. Silverstein. Spectral analysis of networks with random topologies. *SIAM Journal on Applied Mathematics*, 32(2):499–519, 1977.

[6] J. E. Jackson. *A User's Guide to Principal Components*. John Wiley, 1991.

[7] E. Johnson and H. Kargupta. Collective, hierarchical clustering from distributed, heterogeneous data. In *Lecture Notes in Computer Science, volume 1759*, pages 221–244, 1999.

[8] D. Jonsson. Some limit theorems for the eigenvalues of a sample covariance matrix. *Journal of Multivariate Analysis*, 12:1–38, 1982.

[9] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *SIGMOD Workshop on DMKD*, Madison, WI, June 2002.

[10] H. Kargupta, S. Datta, and K. Sivakumar. Random value perturbation: Does it really preserve privacy? In *Technical Report TR-CS-03-25,Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County.*, 2003.

[11] H. Kargupta, K. Liu, and J. Ryan. Random projection and privacy preserving correlation computation from distributed data. In *Technical Report TR-CS-03-24, Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County.*, 2003.

[12] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. Collective data mining: a new perspective towards distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery, Eds: Kargupta, Hillol and Chan, Philip*. AAAI/MIT Press, 2000.

[13] H. Kargupta, H. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. MobiMine: Monitoring the stock market from a PDA. *ACM SIGKDD Explorations*, 3:37–47, 2001.

[14] H. Kargupta, K. Sivakumar, and S. Ghosh. Dependency detection in mobimine and random matrices. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 250–262. Springer, 2002.

[15] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology CRYPTO 2000*, pages 36–54, August 2000.

[16] V. A. Marcenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR — Sbornik*, 1(4):457–483, 1967.

[17] M. L. Mehta. *Random Matrices*. Academic Press, London, 2 edition, 1991.

[18] B. Park, Ayyagari R., and H. Kargupta. A fourier analysis-based approach to learn classifier from distributed heterogeneous data. In *Proceedings of the First SIAM Internation Conference on Data Mining*, Chicago, US, 2001.

[19] B. H. Park and H. Kargupta. Distributed data mining: Algorithms, systems, and applications. In *In Data Mining Handbook*, To be published, 2002.

[20] J. R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81 – 106, 1986.

[21] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[22] B. Schneier. *Applied Cryptography*. John Wiley and Sons, 1995.

[23] J. W. Silverstein. On the weak limit of the largest eigenvalue of a large dimensional sample covariance matrix. *Journal of Multivariate Analysis*, 30(2):307–311, August 1989.

[24] J. W. Silverstein and P. L. Combettes. Signal detection via spectral theory of large dimensional random matrices. *IEEE Transactions on Signal Processing*, 40(8):2100–2105, 1992.

[25] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan. Jam: Java agents for meta-learning over distributed databases. In *Proceedings Third International Conference on Knowledge Discovery and Data Mining*, pages 74–81, Menlo Park, CA, 1997. AAAI Press.

[26] J. F. Traub, Y. Yemini, and H. Woz'niakowski. The statistical security of a statistical database. *ACM Transactions on Database Systems (TODS)*, 9(4):672–679, 1984.

[27] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, CA, July 2002.

[28] Y. Q. Yin, Z. D. Bai, and P. R. Krishnaiah. On the limit of the largest eigenvalue of the large dimensional sample covariance matrix. *Probability Theory and Related Fields*, 78(4):509–521, August 1988.

Detecting Cyber Attacks with Fuzzy Data Mining Techniques

Jonatan Gómez* Dipankar Dasgupta† Fabio Gonzalez‡

Abstract

This paper investigates behavior-based techniques for detecting intrusion/anomalies. We applied techniques based on modeling the normal behavior (positive characterization), i.e., based on a set of normal usage data. Our work attempts to handle the inherent uncertainty in the usage data and in the decision making process using fuzzy sets to describe the input parameter space and the normal behavior patterns. In this way, we integrated efficient fuzzy clustering and recognition techniques that can characterize the abnormal behavior to determine cyber attacks. In particular, we investigated clustering methods that allow data points to belong to more than one cluster with a non-crisp degree (fuzzy membership degree). It can yield an accurate model even in the presence of noise or outliers, can automatically determine the number of clusters, and can yield elastic models that can easily adapt to fluctuations in the monitored system behavior. Also, we used normal usage data to build models for abnormal behavior (negative characterization) in the complement space. Finally, we applied an evolutionary strategy for evolving a set of fuzzy rules in order to build a decision support system for the detection of cyber attacks. Experiments with synthetic and real data sets are performed in order to show the applicability of the proposed approach and to compare with other works reported in the literature.

keywords: Cyber-attacks, Intrusion, Fuzzy, Gravitational, Immune, Evolution

1 Introduction

Nowadays, cyber-terrorism is a potential threat to organizations and countries that have become more dependent on the cyberspace [1]. Securing cyberspace is a challenging task which requires innovative solutions to deal with cyber-terrorism in all its forms and manifestations. One of the cyber terrorism's manifestation is the illegal intrusion into the computer resources of an organization. This illegal access has the objective of extracting, modifying or damaging sensible information of the target organization. Detecting this threat and responding accordingly are the main tasks of intrusion detection tools. The problem of Intrusion Detection (**ID**) has been studied extensively in computer security [2, 3, 4, 5] and has received a lot of attention in machine learning and data mining [6, 7, 8, 9, 10, 11]. The ID problem can be defined as classifying system behavior patterns in two basic categories (normal and abnormal). The following are some of the data mining approaches used in solving the intrusion detection problem:

- *Classical association rules and frequent episodes learning* [8]. These techniques are applied over system audit data to extract consistent and useful patterns of programs and user behavior. These patterns are used to build classifiers that can recognize abnormal behavior.
- *Temporal association rules, in terms of multiple time granularities* [11, 12]. These techniques generate fuzzy and classical temporal association rules.
- *Decision trees* [13]. Normal behavior is used to generate artificial abnormal behavior and a decision tree is generated from these samples.

Because it is hard to build a set of patterns that represents all the abnormal behavior (new intrusion techniques are developed every day around the world), several approaches have tried to build a classifier by using only a set of patterns of the normal behavior [13, 14, 15]. Techniques for building intrusion detection systems by using only normal patterns can be classified as follow:

1. *Negative Characterization*: The normal patterns are used for building a model of the abnormal space;

*Jonatan Gómez is with Computer Science Division, Mathematical Sciences Department, University of Memphis, TN 38152 USA (e-mail:jgomez@memphis.edu) and also with Departamento de Ingeniería de Sistemas, Universidad Nacional de Colombia, Ciudad Universitaria, Bogotá, Colombia.

†Dipankar Dasgupta is with Computer Science Division, Mathematical Sciences Department, University of Memphis, TN 38152 USA (e-mail:dasgupta@memphis.edu).

‡Fabio Gonzalez is with Computer Science Division, Mathematical Sciences Department, University of Memphis, TN 38152 USA (e-mail:fgonzalz@memphis.edu) and also with Departamento de Ingeniería de Sistemas, Universidad Nacional de Colombia, Ciudad Universitaria, Bogotá, Colombia.

for example, by generating a set of rules (detectors) that can recognize abnormal patterns [14, 16, 17].

2. *Positive characterization*: The normal patterns are used for building a model of the normal space (set of rules that defines the normal patterns) [8, 12, 15].

3. *Artificial anomalies generation*: The normal patterns are used for generating artificial abnormal samples and a classifier learning technique is used for generating the classifier [13, 18].

In this paper, we combine the fuzzy gravitational clustering algorithm proposed by Gomez et al. [19] with the fuzzy anomaly identification signatures approach proposed by Gomez et al. [20], and with the evolution of fuzzy rules proposed by Gomez et al. [21, 22], for solving some well-studied intrusion detection problems. In this approach, the gravitational clustering algorithm will generate a good set of clusters that represents the normal behavior (normal space), and the fuzzy cluster analysis will give a better definition of the boundaries between normal and abnormal spaces. In the other hand, the Fuzzy anomaly identification signatures approach will generate a set of good fuzzy rule detectors for the abnormal space. Finally, the fuzzy rules, generated with the evolutionary algorithm, will classify the abnormal behavior in different intrusion types in order to produce an appropriate answer.

The subsequent sections are organized as follows: section 2 briefly describes the gravitational clustering algorithm, the fuzzy anomaly identification signatures and the evolution of fuzzy rules process; section 3 presents the proposed approach to solve some intrusion detection problems; section 4 describes experiments and analysis of results; finally, section 5 draws some conclusions.

2 Background

2.1 Fuzzy Gravitational Clustering (FGC) We developed a robust clustering technique based on the gravitational law and Newton's second motion law [19]. In the gravitational clustering technique, for an n-dimensional data set with N data points, each data point is considered as an object in the n-dimensional space with mass equal to 1. The movement of each point x in the data set due to the gravitational field generated by another point (y), which is randomly chosen, is approximated by using the following dynamic equation:

$$x(t+1) = x(t) + \overrightarrow{d}\frac{G}{\left\|\overrightarrow{d}\right\|^3} \qquad (2.1)$$

where, $\overrightarrow{d} = \overrightarrow{y} - \overrightarrow{x}$ and G is a gravitational constant. This is a computational approximation of the Newtonian movement equation: the velocity of an object is considered equal to zero at any time ($v(t) = 0$), and the time interval length is equal to one ($\triangle(t) = 1$). In this way, the algorithm does not use extra memory for storing the velocity vector of each data point, and it is faster too. It is clear, due to a constant gravitational force, that all the points will be moved to the same position (big crunch) after a large number of iteration. To eliminate this limit effect, we reduced the gravitational constant G in a given proportion in each iteration (decay term: $\triangle(G)$). In each iteration, the algorithm starts to create the clusters by using a disjoint set structure and the distance between objects (after applying the gravitational force). The basic ideas behind applying the gravitational law are:

1. A data point in some cluster exerts a higher gravitational force on a data point in the same cluster than on a data point that is not in the cluster. Then, points in a cluster move in the direction of the center of the cluster. In this way, the proposed technique will determine automatically the clusters in the data set.

2. If some point is a noise point, i.e., does not belong to some cluster, the gravitational force exerted on it for another point is so small that the point is almost immobile. Then, the noise points will not be assigned to any cluster.

In a single iteration, for each data point, another data point is randomly selected, and the two points are moved according to equation 2.1. Clearly, each iteration of the gravitational clustering algorithm is linear on the size of the normal data set used as training (in time and in space).

We used the gravitational clustering algorithm in order to perform anomaly detection [23]. The process is as follows: the gravitational clustering algorithm is run with all or a portion of the normal data patterns in order to generate a set of clusters that represents the normal behavior. Then, each normal pattern is assigned to the closest cluster. After that, some statistical information as cluster radius, average cluster distance, min values and max values per components are calculated. With this statistical information, two fuzzy membership functions are defined for each cluster generated. One fuzzy membership function takes into account the radius and average radius of the cluster (hyper-sphere model), and the other fuzzy membership function takes into account the min and max values per component (hyper-rectangle model). Finally, these two fuzzy membership functions are combined with a *T-norm* operator (average min) to

calculate the membership function of a cluster k, see equation 2.2.

$$F_k(x) = \frac{2 * f_k(x) * g_k(x)}{f_k(x) + g_k(x)} \quad (2.2)$$

Here, $f_k(x)$ is the fuzzy membership generated by the hyper-sphere model, and $g_k(x)$ is the membership function generated by the hyper-rectangle model.

2.2 Fuzzy Anomaly Identification Signatures (FAIS) Forrest et al. [24] developed a negative selection algorithm (NSA) that can be used for discriminating between the normal and abnormal behavior in a computer system. Such negative-selection algorithm can be summarized as follows ([25]):

- Define normal as a collection S of elements in a feature space U, a collection that needs to be monitored. For instance, if U corresponds to the space of states of a system represented by a list of features, S can represent the subset of states that are considered as normal for the system.

- Generate a set R of *rule detectors*, each of which fails to match any string in S. Therefore, each rule detector defines a signature that can be used for identifying anomaly behavior. One simple approach for generating such rule detectors is to generate random rule detectors and discard those that match one or more elements in the normal set S. However, a more efficient approach will try to minimize the number of generated rule detectors while the covering of the abnormal space is maximized.

- Monitor S for changes by continually matching the detectors in R against S. If any detector ever matches, then a change is known to have occurred, as the detectors are designed not to match any of the original strings in S.

Approaches inspired on this idea have been applied successfully to perform anomaly detection on computer networks systems [26, 27, 28]. We modified this idea in order to use a non-binary representation [14], and we extended it to use fuzzy rule detectors instead of crisp rule detectors [20]. In this approach [20], the normal/abnormal space corresponds to the hypercube $[0,1]^n$; therefore, an element x in this space is represented by a vector $(x_1, \ldots, x_n)$, where $x_i \in [0,1]$. For each attribute x_i, a fuzzy division of the real interval $[0,1]$ defines the basic fuzzy sets or linguistic values that such attribute can take. In our experiments, the basic fuzzy sets correspond to a fuzzy division of the real interval $[0,1]$ using triangular and trapezoidal fuzzy membership functions. Figure 1 shows an example of such a division using five basic fuzzy sets representing the linguistic variables *Low*, *Medium-Low*, *Medium*, *Medium-High* and *High*.

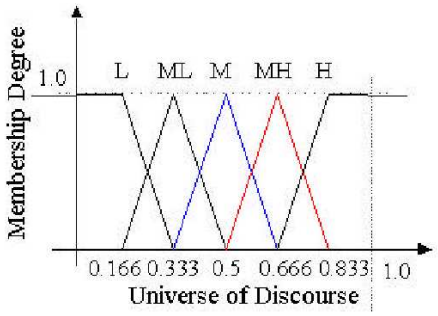

Figure 1: Partition of the interval [0,1] in basic fuzzy sets.

Let $P_i = \{P_{i,1}, P_{i,2}, ..., P_{i,m}\}$ to be the fuzzy sets defined by the attribute i. A fuzzy rule detector has the following structure:

$$\textbf{If } x_1 \in \widehat{T}_1 \wedge \ldots x_n \in \widehat{T}_n \textbf{ then } abnormal$$

Here, $(x_1, \ldots x_n)$ is an element of the normal/abnormal space being evaluated, $\wedge$ is a fuzzy conjunction operator (in our case, min()), T_i is a subset of basic fuzzy sets ($T_i \subseteq P_i$), and $\widehat{T}_i$ is defined as:

$$\widehat{T}_i = \bigcup_{Y \in T_i} Y,$$

where $\bigcup$ corresponds to a fuzzy disjunction operator. We used the addition operator defined as follows:

$$\mu_{A \cup B}(x) = \min\{\mu_A + \mu_B, 1\}.$$

Given a set of rules:

$$\begin{array}{lll} R^1\text{: } \textbf{If} & Cond_1 & \textbf{then abnormal} \\ \vdots & \vdots & \vdots \\ R^m\text{: } \textbf{If} & Cond_m & \textbf{then abnormal,} \end{array}$$

the abnormality degree of a sample x is defined by

$$\mu_{\text{non\_self}}(x) = \max_{i=1,\ldots m} \{Cond_i(x)\}.$$

Here, $Cond_i$ is the condition part of the rule R^i, $Cond_i(x)$ is the fuzzy true value produced by the evaluation of $Cond_i$ in x, and $\mu_{\text{non\_self}}(x)$ represents the degree of membership of x to the abnormal set. Thus, a value close to 0 means that x is normal and a value close to 1 indicates that it is abnormal.

2.3 Evolution of Fuzzy Rules In previous work [21, 22], we evolved fuzzy rules for solving some well known intrusion detection problems . The proposed approach, called **EFRID** (Evolution of Fuzzy Rules for Intrusion Detection), uses an evolutionary algorithm that does not need parameter settings (only population size and number of iterations). In general, an evolved linguistic classifier rule has the following form:

If $x_1 \in S_1$ op_1 $x_2 \in S_2$ ... op_{n-1} $x_n \in S_n$ **Then** *Class* m

where

- x_i ∈[0.0,1.0], is an attribute or linguistic variable;
- S_i ∈L, ML, M, MH, H, is a fuzzy set; and
- op_i ∈AND, OR, is a Fuzzy-Boolean operator.

In this work, the attribute values are normalized in the interval [0,1], and fuzzy sets are defined by the membership functions shown in Figure 1. A fuzzy classifier model can be represented by a set of $m-1$ rules, where m is the number of different classes; that is, m classes are represented by $m-1$ rules. One rule is associated with only one class. For example,

$$\begin{array}{lll} R_1\text{: } \textbf{If} & Cond_1 & \textbf{then class } is\ C_1 \\ \vdots & \vdots & \vdots \\ R_{m-1}\text{: } \textbf{If} & Cond_{m-1} & \textbf{then class } is\ C_{m-1} \end{array}$$

The class that has not associated some rule is considered as default class, i.e., if the set of rules does not classify an element then the element is considered in the default class. In order to classify an unclassified element $(x_1, ..., x_n)$, which is represented by a vector of attributes, the condition part of each rule is evaluated using the membership functions and the fuzzy-set operators. Then, the rule with the highest value in the condition is selected, and the element is classified accordingly to the consequent part of that rule. If all the rules have confidence 0.0 or lower than a fire threshold, the element is classified in the default class. In our experiments, we considered the normal class as default class.

In general, the condition part of a rule corresponds to a logic expression, which can be represented by an expression tree; a linear chromosome with variable length is used to represent this expression tree [22]. A genetic algorithm (GA), which does not require operators probabilities setting, with special operators is applied to evolve the rules. A GA run evolves a rule; so, multiple runs are needed to cover all classes in the training set (excluding the default class). The elements in the training set that belong to the class of the respective run are considered positive examples, and the elements that belong to other classes are considered negative examples. Since there is a different GA run for each class, we do not have to represent the action part of the rule in the chromosome; it only represents the condition part. A fuzzy rule is represented using complete expression trees [22].

3 Combining FGC, FAIS and EFRID

We divided the intrusion detection task in three modules, (see Figure 2):

1. In the first module, a fuzzy deviation level of the monitored parameters from the normal behavior is calculated (using FGC and FAIS).
2. In the second module, the vector of monitored parameters is classified in one of the possible attack class (using EFRID).
3. In the third module, the deviation level and the predicted attack information are combined for determining if the vector is an attack or not.

Because the FGC algorithm gives a fuzzy membership value to the normal space (positive characterization), we use the fuzzy negation operator for determining the deviation level from the normal space calculated by the FGC algorithm:

$$FGC_{dev}(x) = \overline{FGC(x)} = 1 - FGC(x). \qquad (3.3)$$

The fuzzy deviation level of a vector of monitored parameters is calculated as the minimum fuzzy deviation level due to the FGC and the FAIS algorithms.

$$DEV(x) = min\left\{FAIS(x), FGC_{dev}(x)\right\}. \qquad (3.4)$$

We used the EFRID algorithm for generating a fuzzy classifier for the known attacks (the normal class is considered as default class). In this way, the second module calculates an abnormality value of a vector of monitored parameters (the fuzzy rule value).

In the third module, the information generated by the deviation module is combined with the information provided by the classification module. We use the multiplication fuzzy-and operator in this module.

$$FIDS(x) = DEV(x) * EFRID(x). \qquad (3.5)$$

There are two elements that define the cost function of an anomaly detection system: the false alarm rate (**FA**), the system produces an alarm in normal conditions, and the detection rate (**DR**), the system detects

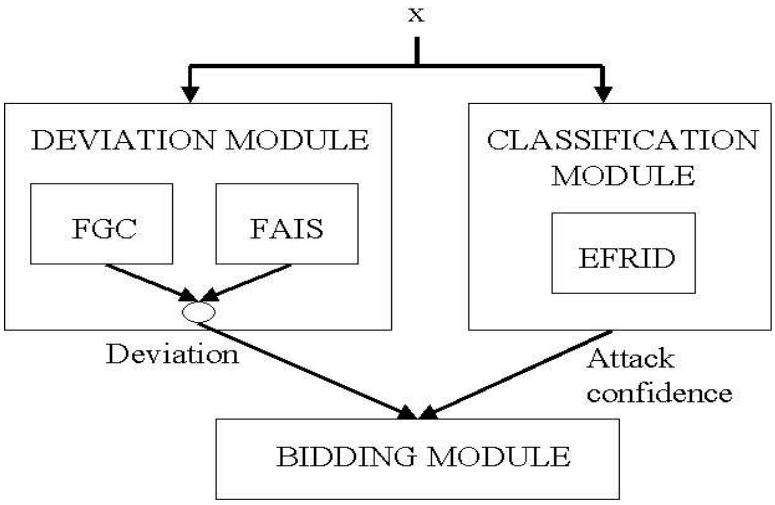

Figure 2: Combining Fuzzy Gravitational Clustering (FGC), with Fuzzy Anomaly Identification Signatures (FAIS) and the Evolution of Fuzzy Rules for Intrusion Detection (EFRID).

an attack. A good intrusion detection system is one that has low FA and high DR. In general, positive characterization techniques generate a low false alarm and detection rates, whereas negative characterization techniques generate a high false alarm and detection rates (see Figure 3).

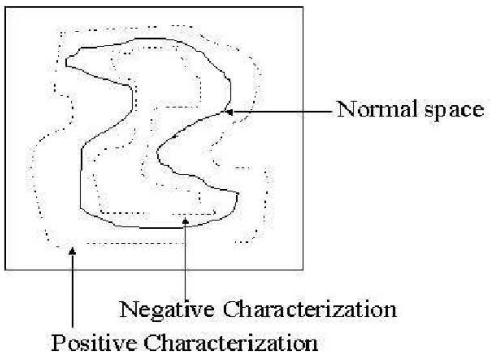

Figure 3: Positive and Negative Characterization Techniques Comparison

We combined the FGC, FAIS, and EFRID modules in order to achieve two goals with FIDS. Our first goal is to provide a better characterization of the normal-abnormal space by combining FGC and FAIS. In this way we want that FIDS keeps the low false alarm rate provided by the FGC algorithm (or similar) while reaching the detection rate provided by the FAIS technique. In this way, the double fuzzyfication (FGC and FAIS) of the normal-abnormal boundaries will provide a more useful definition of normalcy. Our second goal is to provide a good classification of the monitored behavior as abnormal with the EFRID module. This characterization will provide a good strategy for answering correctly to possible attacks.

4 Experimental Results

In order to determine the performance of the proposed approach, experiments were conducted with two different data sets as shown in Table 1. We used 50% of the data set, randomly selected, as training set and the remaining 50% as testing set. We generated the training and testing data sets in such a way that they have the same number of records from each class. Also, we generated a normal training data set (a set with only normal samples from the training set). We used the normal training data set in the FGC and FAIS approaches, while we used the full training data set in the EFRID approach. In order to compare the performance of the proposed approach, we generated a ROC curve [29] for each algorithm used.

Table 1: Data sets used for experimentation

Data Set	Number of Samples	
	Normal	Abnormal
Simple ML	1000	150
KDD-Cup 99	95278	396743

4.1 Parameters Setting We defined a set of parameters values for each of the techniques used here: FGC, FAIS, and EFRID.

4.1.1 FGC Because it is possible to use a portion of the normal training data set in the gravitational clustering algorithm for generating a good set of clusters, see [19], only a 10% of the training data set was used by the FGC algorithm. The reported results were obtained using the following parameter values: gravitational force, $G = 1e-4$; gravitational force decreasing factor, $\triangle(G) = 0.01$; clusters size, $\alpha = 0.01$; epsilon, $\varepsilon = 1e-6$; and maximum number of iterations, $M = 100$.

4.1.2 FAIS The reported results were obtained with the following set of parameter values: population size, $PS = 200$; number of samples per individual, $SS = 400$; number of fuzzy sets per attribute, $LV = 5$ (as shown in 1); and number of iterations, $M = 1000$.

4.1.3 EFRID The reported results were obtained with the following set of parameter values: population size, $PS = 200$; number of samples per individual, $SS = 400$; number of fuzzy sets per attribute, $LV = 5$ (as shown in 1); and number of iterations, $M = 1000$.

4.2 KDD Cup 99

4.2.1 Description This data set is a version of the 1998 DARPA intrusion detection evaluation data set prepared and managed by MIT Lincoln Labs[1]. Experiments were conducted on the 10% that is available at the University of Irvine Machine Learning repository [2]. Forty-two attributes, that usually characterize network traffic behavior, compose each record of the 10% data set (22 of them numerical). Also, the number of records in the 10% is huge (492021).

We generated a reduced version of the 10% data set including only the numerical attributes, i.e., the categorical attributes were removed from the data set. Therefore, the reduced 10% data set is composed of 33 attributes. The attributes were normalized between 0 and 1 using the maximum and minimum values found.

4.2.2 Results Figure 4 shows the ROC curves generated by the FGC, FAIS, and FIDS.

As expected, while FGC reaches low FA and DR values (FGC is a positive characterization technique), FAIS reaches a high FA and DR values, (FAIS is a negative characterization technique). Clearly, FIDS shows the performance desired: It keeps the low FA rate provided by the FGC algorithm while reaches the DR provided by the FAIS algorithm. FIDS reaches a DR of 98.7%, while it keeps a low FA rate ($< 1\%$). Table 2 compares the performance reached by FIDS against some results reported in the literature. As shown FIDS performs very well compared with such methods. Also, the reduction of FA rate compared with EFRID is significant, (6%).

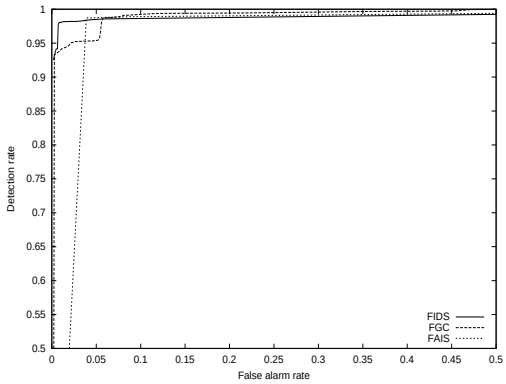

Figure 4: Roc curve generated for the KDD Cup 99 data set

Approach	FA%	DR%
FIDS	**0.9**	**98.70**
EFRID [21]	7.0	98.95
RIPPER-AA[13]	2.02	94.26
SMARTSIFTER[30]	-	82.0

Table 2: Comparative Performance

The EFRID algorithm generated a set of four fuzzy rules, one for each main attack classification: Denial of Service (DoS), Probe (PRB), Remote to Local (R2L) and User to Root (U2R), (see [21]). The rules generated by EFRID are shown in Figure 5.

Figures 6 and 7 show the ability of the EFRID algorithm for extracting useful fuzzy rules from the training data set. In these figures, the data set is shown according to the attributes used in the condition parts of the fuzzy rules. Figure 6 shows the data set according to the attributes used in the condition part of the DoS fuzzy rule. As shown, almost all the DoS records have a *count* value higher than 0.2. Since we defined a fuzzy division of each attribute as shown in Figure 1, these values (> 0.2) can be easily characterized as *not count*.

[1] Mit Lincoln labs. 1999 Darpa Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/index.html

[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

if count is not low or same_srv_rate is low
then record-type is DoS
if dst_host_rerror_rate is not low
then record-type is PRB
if dst_host_same_src_port_rate is not low and count is not high **then** record-type is R2L
if src_bytes is medium-high
then record-type is U2R

Figure 5: Fuzzy rules evolved by EFRID

Similarly, the attribute *same_srv_rate* can be used to characterize many of the DoS samples that were not identified by the *not count* part of the rule. Therefore, the EFRID fuzzy rule characterizes the DoS attack very well. In the same way, figure 7 shows the data set (without the DoS records) according to the attributes in fuzzy rules PRB, and U2R. As shown these fuzzy rules are useful for discriminating among different attacks.

Finally, the classification accuracy reached by FIDS is shown in table 3 along with the performance of the winner group in the KDDCup' 99 contest [31]. However, these results are not comparable because different data sets were used for testing. So, this information is for reference only. The numbers between parenthesis indicate the percentage of samples from the class that were classified as abnormal.

CLASS	FIDS	WINNER ENTRY
Normal	98.70%	94.50%
U2R	0.0% (17.24%)	13.20%
R2L	12.33% (12.5%)	8.40%
DoS	95.50% (98.64%)	97.10%
PRB	20.69% (76.0%)	83.30%

Table 3: Classification Accuracy

4.3 Multi-Level Network Traffic Data

4.3.1 Description The Cougaar Intrusion Detection System (CIDS) is a system that allows detection of intrusions in a computer network. Written on Java language and using the Cougaar agent architecture[3], this tool was developed by the Intelligent Security System Research Lab (ISSRL) at the University of Memphis. CIDS allows the monitoring at different levels (process, user, network) of several computer network host parameters. Table 4 shows the twenty-one parameters monitored in CIDS.

[3] http://www.cougaar.org/

Data of a computer host (target) behavior (under attack and in normal conditions) were collected during several hours in intervals of time of 10 seconds. Two attacks were performed over the target host, a PRB attack using the *nmap* scan tool and U2R by using a secure shell hacking tool (*ssh*).

4.3.2 Results The proposed approach (FIDS) was able to discriminate the normal behavior from the abnormal behavior in all the cases. Moreover, the classification of the possible attack was almost perfect. In 99.97% of the testing cases, FIDS was able to classify correctly the attacks launched against the computer host. The set of fuzzy rules generated in this case were:

if UsedSwapRam is not high and RemotePacketsSent is low **then** record-type is ssh-hack

if RemotePacketsSent is not low **then** record-type is nmap

Figure 8 shows the data set according to the attributes used in the condition part of the fuzzy rules generated by EFRID. As shown, the fuzzy rules generated by EFRID are very good for differentiating among the abnormal classes and also for differentiating among the normal and the abnormal classes.

5 Conclusions

This paper combines a fuzzy gravitational clustering algorithm [19] with the fuzzy anomaly identification signatures approach [20], and the evolution of fuzzy rules proposed [21, 22], for solving some well-studied intrusion detection problems. It is shown that combining positive characterization techniques with negative characterization techniques can help in the reduction of the FA rate while the DR is kept high. Also, It is shown that evolution of fuzzy rules for classifying the monitored behavior can help in the determination of the appropriate answer for specific attacks when they are detected by the deviation module (FGC and FAIS techniques).

The experimental results showed that fuzzy analysis can improve the performance of an intrusion detection system.

6 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (F30602-00-2-0514).

References

[1] J. Schwartz, "Cyberterrorists sharpening their tools for online war fare.," *International Herald Tribune*, 2003.

[2] E. Amoroso, *Intrusion Detection.* Intrusion.net Books, 1999.

[3] R. Heady, G. Luger, A. Maccabe, and M. Sevilla, "The Architecture of a Network-level Intrusion Detection System," Tech. Rep. CS90-20, University of New Mexico, Albuquerque.

[4] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," Tech. Rep. 99-15, Chalmers University of Technology, Dept. of Computer Engineering, Mar. 2000.

[5] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, "State of the Practice of Intrusion Detection Technologies," Tech. Rep. CMU/SEI99-TR-028, ESC-99-028, Carnegie Mellon, Software Engineering Institute, Pittsburgh, 1999.

[6] J. Sundar, D. Garcia-Fernandez, E. Spafford, and D. Zamboni, "An Architecture for Intrusion Detection using Autonomous Agents," Tech. Rep. 98/05, Purdue University, 1998.

[7] M. Crosbie, "Applying Genetic Programming to Intrusion Detection," in *AAAI 1995 Fall Symposium Series*, Nov. 1995.

[8] W. Lee, S. Stolfo, and K. Mok, "Mining Audit Data to Build Intrusion Detection Models," in *International Conference Knowledge Discovery and Data Mining (KDD'98)*, pp. 66–72, 1998.

[9] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *IEEE Symposium on Security and Privacy*, pp. 120–132, 1999.

[10] W. Lee, S. Stolfo, and K. Mok, "Mining in a data-flow environment: Experience in network intrusion detection," in *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)* (S. Chaudhuri and D. Madigan, eds.), pp. 114–124, 1999.

[11] Y. Li, N. Wu, S. Jajodia, and X. S. Wang, "Enhancing profiles for anomaly detection using time granularities."

[12] S. Bridges and R. Vaughn, "Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection," in *Twenty Third National Information Security Conference*, Oct. 2000.

[13] W. Fan, W. Lee, M. Miller, S. Stolfo, and P. Chan, "Using artificial anomalies to detect unknown and known network intrusions," in *Proceedings of the first IEEE International conference on Data Mining*, 2001.

[14] D. Dasgupta and F. Gonzalez, "An immunity-based technique to characterize intrusions in computer networks," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 281–291, 1999.

[15] S. Hofmeyr, A. Somayaji, and S. Forrest, "Intrusion Detection Using Sequences of Systems Call," *Computer Security*, no. 6, pp. 151–180, 1998.

[16] F. Gonzalez and D. Dasgupta, "An immunogenetic technique to detect anomalies in network traffic," in *Proceedings of the genetic and evolutionary compuation conference, GECCO 2002*, pp. 1081–1088, Morgan Kaufman Publishers, 2002.

[17] D. Dasgupta and S. Forrest, "An anomaly detection algorithm inspired by the immune system," *Artificial immune systems and their applications*, pp. 262–277, 1999.

[18] F. Gonzalez and D. Dasgupta, "Neuro-immune and self-organising map approaches to anomaly detection: A comparison," in *!st International Conference on Artificial Immune Systems*, Sept. 2002.

[19] J. Gomez, D. Dasgupta, and O. Nasraoui, "A New Gravitational Clustering Algorithm," in *To appear in the Proceedings of the Third SIAM International Conference on Data Mining 2003*, 2003.

[20] J. Gomez, F. Gonzalez, and D. Dasgupta, "An Immuno-Fuzzy Approach to Anomaly Detection," in *To appear in the Proceeding of the IEEE International Conference on Fuzzy Systems FUZZIEEE 2003*, 2003.

[21] J. Gomez and D. Dasgupta, "Evolving Fuzzy Classifiers for Intrusion Detection," in *Proceedings of the 2002 IEEE Workshop on Information Assurance*, June 2002.

[22] J. Gomez, D. Dasgupta, O. Nasraoui, and F. Gonzalez, "Complete expression trees for evolving fuzzy classifiers systems with genetic algorithms," in *Proceedings of NAFIPS-FLINT joint conference*, June 2002.

[23] J. Gomez and D. Dasgupta, "Combining Gravitational Clustering and Fuzzy Cluster Analysis For Anomaly Detection." Submited to the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03).

[24] S. Forrest, A. Perelson, L. Allen, and R. Cherukury, "Self-nonself discrimination in a computer," in *Proceedings of the IEEE Symp. on research in security and privacy*, 1994.

[25] D. Dasgupta, "An overview of artificial immune system and theri applications," *Artificial immune systems and their applications*, pp. 3–23, 1999.

[26] D. Dasgupta. Springer-Verlag, 1999.

[27] S. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evolutionary Computation*, vol. 8, no. 4, pp. 443–473, 2000.

[28] J. Kephart, "A biologically inspired immune system for computers," in *Procedings of Artificial Life, (Cambrige M,A)*, pp. 130–139, 1994.

[29] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of 15th intenational conference on machine learning*, p. 445;453, 1998.

[30] K. Yamanishi, J. Takeuchi, and G. Williams, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 320–324, 2000.

[31] "Results of the kdd' 99 classifier learning contest.." http://www-cse.ucsd.edu/users/elkan/clresults.html.

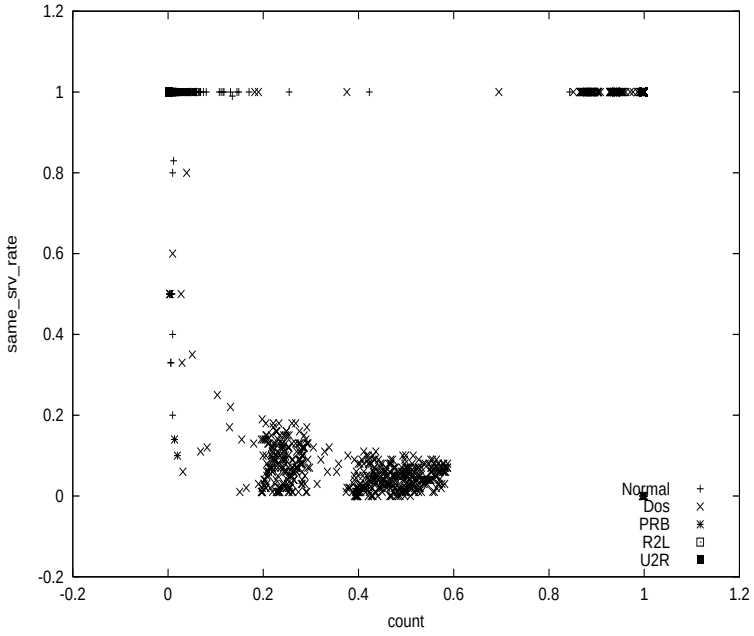

Figure 6: Projection of the KDDCup data set using the DoS classification attributes.

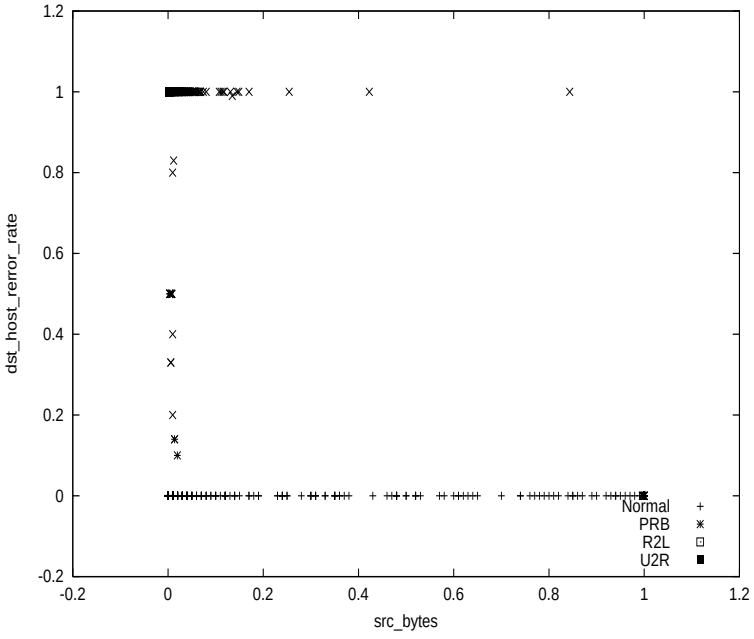

Figure 7: Projection of the KDDCup data set using the PRB and U2L classification attributes.

PARAMETER	PARAMETER	PARAMETER
LOCAL_SENT_BYTES	REMOTE_RECEIVED_PACKETS	PROCESSES_ZOMBIE
LOCAL_RECEIVED_BYTES	PROCESSES	USED_PHYSICAL_RAM
LOCAL_SENT_PACKETS	PROCESSES_ROOT	USED_SWAP_RAM
LOCAL_RECEIVED_PACKETS	PROCESSES_USER	LOGINS
REMOTE_SENT_BYTES	PROCESSES_BLOCKED	FAILED_LOGINS
REMOTE_RECEIVED_BYTES	PROCESSES_RUNNING	REMOTE_LOGINS
REMOTE_SENT_PACKETS	PROCESSES_WAITING	CPU_USERS

Table 4: Monitored parameters with CIDS (Cougaar based Intrusion Detection System).

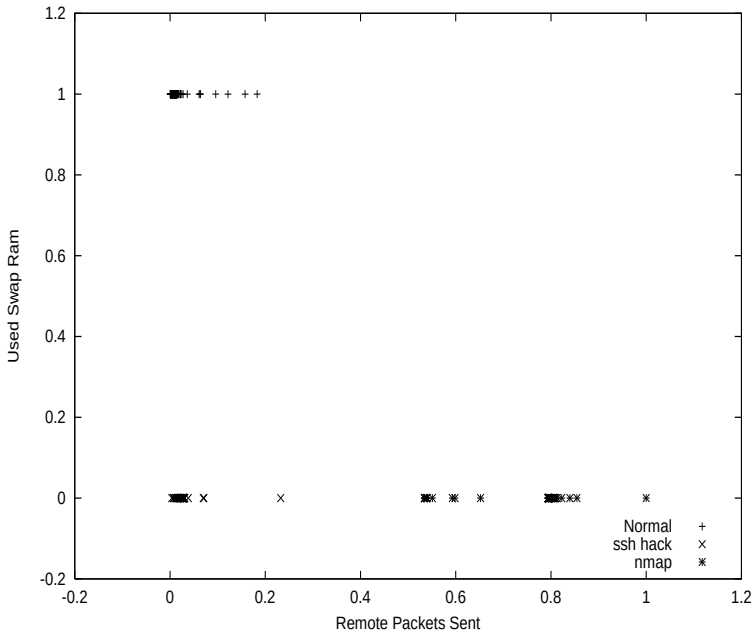

Figure 8: Attributes used in the EFRID fuzzy rules for classifying the CIDS data set.

Near Linear Time Detection of Distance-Based Outliers and Applications to Security

Stephen D. Bay[1] **and Mark Schwabacher**[2]

[1]Institute for the Study of Learning and Expertise
2164 Staunton Court
Palo Alto, CA 94306
sbay@apres.stanford.edu

[2]NASA Ames Research Center
Computational Sciences Division
MS 269-3, Moffet Field, CA 94035
Mark.A.Schwabacher@nasa.gov

Abstract

Many automated systems for detecting threats are based on matching a new database record to known attack types. However, this approach can only spot known threats and thus researchers have also begun to use unsupervised approaches based on detecting outliers or anomalous examples. A popular method of finding these outliers is to use the distance to an example's k nearest neighbors as a measure of unusualness. However, existing algorithms for finding distance-based outliers have poor scaling properties, making it difficult to apply them to large datasets typically available in security domains. In this paper, we propose modifications to a simple, but quadratic, algorithm for finding distance-based outliers, and show that it achieves near linear time scaling allowing it to be applied to real data sets with millions of examples and many features.

Keywords: outlier detection, anomaly detection, nearest neighbors, aviation security

1 Introduction

Detecting threats by analyzing the examples in a database is an important problem in security domains. For example, researchers in data mining are developing algorithms to detect computer intrusions from audit records [21, 10, 19]. The U.S. Federal Aviation Administration developed the Computer Assisted Passenger Pre-screening System (CAPPS) [8, 11, 25], which screens airline passengers on the basis of their flight records and flags individuals for additional checked baggage screening.

One approach to addressing this task is developing sophisticated models of a threat and how they manifest themselves in a data set record. For example, although the exact details of CAPPS are not published, it is thought to assign higher threat scores to cash payments [25]. However, explicit threat models can only capture known attack types. Another approach is based on outlier or anomaly detection where one looks for unusual examples that appear suspicious and may require additional screening [20, 10, 22].

Outlier detection has a long history in statistics [3, 14], but has largely focussed on univariate data with a known distribution. These two limitations have restricted the ability to apply these types of methods to large real-world databases which typically have many different fields and have no easy way of characterizing the multivariate distribution of examples. Other researchers, beginning with the work by Knorr and Ng [17], have taken a non-parametric approach and proposed using an example's distance to its nearest neighbors as a measure of unusualness [2, 23, 18, 10]. Eskin et al. [10], and Lane and Brodley [20] applied distance-based outliers to detecting computer intrusions from audit data.

Although distance is an effective non-parametric approach to detecting outliers, the drawback is the amount of computation time required. Straightforward algorithms, such as those based on nested loops, typically require $O(N^2)$ distance computations. This quadratic scaling means that it will be very difficult

to mine outliers as we tackle increasingly larger data sets. This is a major problem for security screening databases where there are often millions of records. For example, U.S. airlines carry over 600 million passengers per year[8].

Recently, researchers have presented many different algorithms for efficiently finding distance-based outliers. These approaches vary from spatial indexing trees to partitioning of the feature space with clustering algorithms [23]. The main goal is developing algorithms that scale to large real data sets.

In this paper, we show that one can modify a simple algorithm based on nested loops, which would normally have quadratic scaling behavior, to yield near linear time mining on real, large, and high-dimensional data sets. Our goal is to develop algorithms for mining distance-based outliers that can feasibly be applied to help detect threats in large security data sets. Specifically, our contributions are:

- We show that an algorithm based on nested loops in conjunction with randomization and a simple pruning rule has near linear time performance on many large real data sets. Previous work reported quadratic performance for algorithms based on nested loops [17, 18, 23].
- We demonstrate that our algorithm scales to real data sets with millions of examples and many features, both continuous and discrete. To our knowledge we have run our algorithm on the largest reported data sets to date and obtained among the best scaling results for real data sets. Other work has reported algorithms with linear time mining but only for low-dimensional problems (less than 5) [17, 18] or have only tested the scaling properties on simple synthetic domains.
- We analyze why our algorithm performs so well. Under certain conditions, the results suggest that the time to process non-outliers, which are the large majority of points, does not depend on the size of the data set.
- We apply our algorithm to an airline passenger database and we discuss some limitations of our approach on this database.

The remainder of this paper is organized as follows. In the next section, we review the notion of distance-based outliers and present a simple nested loop algorithm that will be the focus of this paper. In Section 3, we show that although our simple algorithm has poor worst case scaling properties, for many large, high-dimensional, real data sets the actual performance is extremely good and is close to linear. In Section 4, we analyze our algorithm and attempt to explain the performance with an average case analysis. In Section 5, we present examples of discovered outliers to give the readers a qualitative feel for how the algorithm works on real data. Finally, we conclude this paper by discussing limitations and directions for future work.

2 Distance-Based Outliers

A popular method of identifying outliers is by examining the distance to an example's nearest neighbors [23, 18, 17, 2]. In this approach, one looks at the local neighborhood of points for an example typically defined by the k nearest examples (also known as neighbors). If the neighboring points are relatively close, then the example is considered normal; if the neighboring points are far away, then the example is considered unusual. The advantages of distance-based outliers are that no explicit distribution needs to be defined to determine unusualness, and that it can be applied to any feature space for which we can define a distance measure.

Given a distance measure on a feature space, there are many different definitions of distance-based outliers. Three popular definitions are

1. Outliers are the examples for which there are less than p other examples within distance d [17, 18].
2. Outliers are the top n examples whose distance to the kth nearest neighbor is greatest [23].
3. Outliers are the top n examples whose average distance to the k nearest neighbors is greatest [2, 10].

There are several minor differences between these definitions. The first definition does not provide a ranking and requires specifying a distance parameter d. Ramaswamy et al. [23] argue that this parameter could be difficult to determine and may involve trial and error to guess an appropriate value. The second definition only considers the distance to the kth neighbor and ignores information about closer points. Finally, the last definition accounts for the distance to each neighbor but is slower to calculate than definition 1 or 2. However, all of these definitions are based on a nearest neighbor density estimate [12] to determine the points in low probability regions which are considered outliers.

Researchers have tried a variety of approaches to find these outliers efficiently. The simplest are those using nested loops [17, 18, 23]. In the basic version one compares each example with every other example to determine its k nearest neighbors. Given the

neighbors for each example in the data set, simply select the top n candidates according to the outlier definition. This approach has quadratic complexity as we must make all pairwise distance computations between examples.

Another method for finding outliers is to use a spatial indexing structure such as a KD-tree [4], R-tree [13], or X-tree [5] to find the nearest neighbors of each candidate point [17, 18, 23]. One queries the index structure for the closest k points to each example, and as before one simply selects the top candidates according to the outlier definition. For low-dimensional data sets this approach can work extremely well and potentially scales as $N \log N$ if the index tree can find an example's nearest neighbors in $\log N$ time. However, index structures break down as the dimensionality increases. For example, Breunig et al. [7] used a variant of the X-tree to do nearest neighbor search and found that the index only worked well for low-dimensions, less than 5, and performance dramatically worsened for just 10 or 20 dimensions. In fact, for high-dimensional data they recommended sequential scanning over the index tree.

A few researchers have proposed partitioning the space into regions and thus allowing faster determination of the nearest neighbors. For each region, one stores summary statistics such as the minimum bounding rectangle. During nearest neighbor search, one compares the example to the bounding rectangle to determine if it is possible for a nearest neighbor to come from that region. If it is not possible, all points in the region are eliminated as possible neighbors. Knorr and Ng [17] partition the space into cells that are hyper-rectangles. This yields a complexity linear in N but exponential in the number of dimensions. They found that this cell based approach outperformed the nested loop algorithm, which is quadratic in N, only for four or fewer dimensions. Others use a linear time clustering algorithm to partition the data set [23, 10]. With this approach, Ramaswamy et al. demonstrated much better performance compared with the nested loop and indexing approaches on a low-dimensional synthetic data set. However, their experiments did not test how it would scale on larger and higher-dimensional data.

Finally, a few researchers have advocated projections to find outliers. Aggrawal and Yu [1] suggest that because of the curse of dimensionality one should focus on finding outliers in low-dimensional projections. Angiulli and Pizzuti [2] project the data in the full feature space multiple times onto the interval [0,1] with Hilbert space filling curves. Each successive projection improves the estimate of an example's outlier score in the full-dimensional space. Their initial scaling results are promising, and appear to be close to linear, however they have reported results on only two synthetic domains.

In this paper, we show that the simplest type of algorithm based on nested loops in conjunction with randomization and a pruning rule gives state-of-the-art performance. Table 1 shows our variation of the nested loop algorithm in more detail. The function distance computes the distance between any two examples using, for example, Euclidean distance for continuous features and Hamming distance for discrete features. The score function can be any monotonically decreasing function of the nearest neighbor distances such as the distance to the kth nearest neighbor, or the average distance to the k neighbors.

The main idea in our nested loop algorithm is that for each example in D we keep track of the closest neighbors found so far. When an example's closest neighbors achieve a score lower than the cutoff we remove the example because it can no longer be an outlier. As we process more examples, the algorithm finds more extreme outliers and the cutoff increases along with pruning efficiency.

Note that we assume that the examples in the data set are in random order. The examples can be put into random order in linear time and constant main memory with a disk-based randomization algorithm. One repeatedly shuffles the data set into random piles and then concatenates them in random order.

In the worst case, the performance of the algorithm is very poor. Because of the nested loops, it could require $O(N^2)$ distance computations and $O(N/blocksize * N)$ data accesses.

3 Experiments on Scaling Performance

In this section, we examine the empirical performance of the simple algorithm on several large real data sets. The primary question we are interested in answering is "How does the running time scale with the number of data points for large data sets?" In addition, we are also interested in understanding how the running time scales with k, the number of nearest neighbors.

To test our algorithm we selected the five real and one synthetic data set summarized in Table 2. These data sets span a range of problems and have very different types of features. We describe each in more detail.

- *Corel Histogram.* Each example in this data set encodes the color histogram of an image in a collection of photographs. The histogram has 32

Table 1: A simple algorithm for finding distance-based outliers. Lowercase variables represent scalar values and uppercase variables represents sets.

```
Procedure: Find Outliers
Input: k, the number of nearest neighbors; n, the number of outliers to return; D, a set of examples
in random order.
Output: O, a set of outliers.
Let maxdist(x, Y) return the maximum distance between x and an example in Y.
Let Closest(x, Y, k) return the k closest examples in Y to x.
begin
1.      c ← 0       // set the cutoff for pruning to 0
2.      O ← ∅       // initialize to the empty set
3.      while B ← get-next-block(D) {       // load a block of examples from D
4.          Neighbors(b) ← ∅ for all b in B
5.          for each d in D {
6.              for each b in B, b ≠ d {
7.                  if |Neighbors(b)| < k or distance(b,d) < maxdist(Neighbors(b),b) {
8.                      Neighbors(b) ← Closest(b,Neighbors(b) ∪ d, k)
9.                      if score(Neighbors(b),b) < c {
10.                         remove b from B
11.                     } } } }
12.         O ← Top(B ∪ O,n)      // keep only the top n outliers
13.         c ← min(score(o)) for all o in O       // the cutoff is the score of the weakest outlier
14.     }
15.     return O
end
```

bins corresponding to eight levels of hue and four levels of saturation.

- *Airline Passenger.* We obtained 90 days' worth of passenger data from a major U.S. airline. This database includes the information that each passenger provided to the airline (or to a travel agent) when buying an airline ticket. The passenger data is best seen as a relational database. For example, each record can contain multiple passengers traveling together, and each group of passengers can have multiple flight segments. For the experiments described in this paper, we flattened one day's worth of data to create a table in which each row represents one passenger flight segment. We selected 15 fields from this flattened table which we believe could be useful for passenger screening. We intend to scale up to 90 days' worth of data and to develop anomaly detection algorithms that operate directly on relational databases without first flattening them (see Section 6).

- *KDDCUP 1999.* The KDDCUP data contains a set of records that represent connections to a military computer network where there have been multiple intrusions by unauthorized users. The raw binary TCP data from the network has been processed into features such as the connection duration, protocol type, number of failed logins, and so forth.

- *Census.* This data set contains the responses from the 1990 decennial Census in the United States. The data has information on both households and individuals. We divided the responses into two tables, one that stores household records and another that stores person records, and treated each table as its own data set. Both the Household and Person data sets have a variety of geographic, economic, and demographic variables. Our data comes from the 5% State public use microdata samples and we used the short variable list [24]. In total, the 5% State sample contains about 5.5 million household and 12.5 million person records. For our experiments we used a maximum of 5 million records for each data set.

- *Normal 30D.* This is a synthetic data set generated from a 30-dimensional normal distribution centered on the origin with a covariance matrix equal to the identity matrix.

We obtained the data sets Corel Histogram and KDDCup 1999 from the UCI KDD Archive [15] and the census data from the IPUMS repository [24].

Table 2: Description of Data Sets

Data Set	Features	Cont.	Examples
Corel Histogram	32	32	68,040
Airline Passenger	15	3	439,381
KDDCup 1999	42	34	4,898,430
Household 1990	23	9	5,000,000
Person 1990	55	20	5,000,000
Normal 30D	30	30	1,000,000

We preprocessed the data by normalizing all continuous variables to the range [0,1] and converting all categorical variables to an integer representation. We then randomized the order of examples in the data sets. Randomizing a file can be done in $O(N)$ time and constant main memory with a disk-based shuffling algorithm as follows: Sequentially process each example in the data set by randomly placing it into one of n different piles. Recombine the piles in random order and repeat this process a fixed number of times.

We ran our experiments on a lightly loaded Pentium 4 computer with a 1.5 GHz processor and 1GB RAM running Linux. We report the wall clock time, the time a user would have to wait for the output, in order to measure both CPU and I/O time. The reported times do not include the time needed for the initial randomization of the data set and represent one trial. Preliminary experiments indicated that alternate randomizations did not have a major effect on the running time. To measure scaling, we generated smaller data sets by taking the first n samples of the randomized set. Unless otherwise noted, we ran experiments to return the top 30 anomalies with $k = 5$, a block size ($|B|$) of 1000 examples, and we used the average distance to the nearest k neighbors as the score function.

Our implementation of the algorithm was written in C++ and compiled with gcc version 2.96 with the -O3 optimization flag. We accessed examples in the data set sequentially using standard **iostream** functions and we did not write any special routines to perform caching. The total memory footprint of the executing program was typically less than 3 MB.

Figure 1 shows the total time taken to mine outliers on the six data sets as the number of examples varied. Note that both the x and y axes are in a logarithmic scale. Each graph shows three lines. The bottom line represents the theoretical time necessary to mine the data set given a linear algorithm based on the running time for $N = 1000$. The middle line shows the actual running times of our system. Finally, the top line shows the theoretical time needed assuming a quadratic algorithm based on scaling the running time for $N = 1000$.

These results show that our simple algorithm gives extremely good scaling performance that is near linear time. The scaling properties hold for data sets with both continuous and discrete features and the properties hold over several orders of magnitude of increasing data set size. The plotted points typically follow a straight line on the log-log graph which means that the relationship between the y and x axis variables is of the form $y = ax^b$ or $\log y = \log a + b \log x$, where a and b are constants. Thus, the algorithm scales with a polynomial complexity with an exponent equal to the slope of the line. Table 3 presents for each data set the slope of a regression line fit to the points in Figure 1. The algorithm obtained a polynomial scaling complexity with exponent varying from 1.13 to 1.32.

Table 3: Slope b of the regression fit relating $\log t = \log a + b \log N$ (or $t = aN^b$) where t is the total time (CPU + I/O), N is the number of data points, and a is constant factor.

Data Set	slope
Corel Histogram	1.13
Airline Passenger	1.20
KDDCup 1999	1.13
Household 1990	1.32
Person 1990	1.16
Normal 30D	1.15

One exception to the straight line behavior is the last point plotted for the Airline Passenger data set. Up to 100,000 examples the running time follows a straight line very closely with $b = 1.08$, however, the last point at $N = 439,000$ represents a large increase in time. We believe this is caused by the way we flattened the original relational database. For example, if there are two tables X and Y, with each example in X pointing to several different objects in Y, our flattened database will have examples with form (X_1, Y_1), (X_1, Y_2), (X_1, Y_3), (X_2, Y_4), ... and so forth. As it is likely that the closest neighbors of (X_1, Y_1) will be the examples (X_1, Y_2) and (X_1, Y_3) our algorithm may have to scan the entire data set until it finds them to obtain a low score. For small random subsets, it is unlikely that the related examples would be present and so this does not affect scaling performance. Flattening destroys the independence between examples which we will see in Section 4 is necessary for good performance.[1]

[1]We are also considering alternative problem formulations

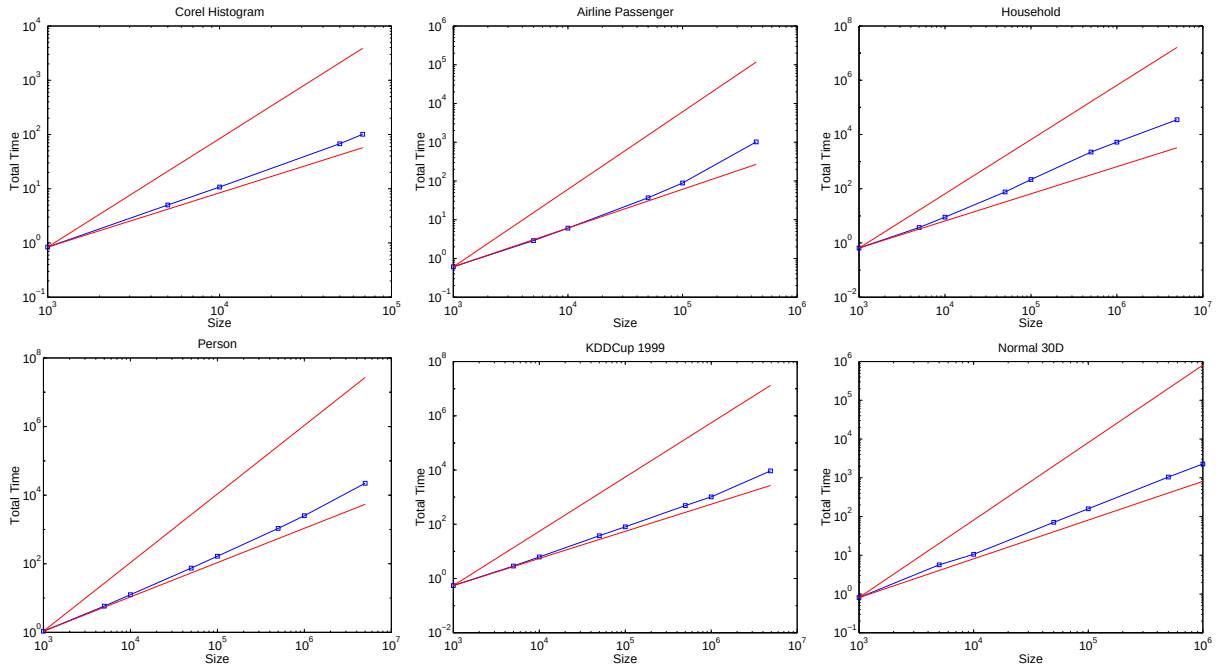

Figure 1: Total time (CPU and I/O) taken to mine outliers as N, the number of points, increases. The top and bottom lines represent the theoretical time taken by a quadratic and linear algorithm based on scaling the observed time at $N = 1000$.

We also examined how the total running time scales with k, the number of neighbors and the results for Normal 30D and Person ($N = 1,000,000$) are shown in Figure 2. The bottom line represents the observed time; the curved top line represents the time assuming linear scaling based on the timing results for $k = 5$. In these graphs, the y axis is logarithmic and the x axis is linear which means that a straight line indicates that the relationship between the y and x axis variables is of the form $y = ae^{bx}$ or $\log y = \log a + bx$ where a and b are constants. This relationship suggests that the running time scales exponentially with k. However, the empirical value of b as determined by a regression fit is very small. For Normal 30D $b = 0.0163$ and for Person $b = 0.0135$. Practically, the observed scaling performance is much better than linear for $k \leq 100$, mainly because of the large fixed computation costs unrelated to k.

4 Analysis of Scaling Time

In this section, we explain with an average case analysis why randomization in conjunction with pruning performs well, especially when much of the past literature reported that nested loop designs were extremely slow because of the $O(N^2)$ distance computations. In particular, both Knorr and Ng [17] and Ramaswamy et al. [23] implemented versions of the nested loop algorithm and reported quadratic performance.

Consider the number of distance computations needed to process an example $\mathbf{x}$. For now we assume that we are using outlier definition 2, rather than definition 3 which we used in our experiment, for ease of analysis. With this definition an outlier is determined by the distance to its kth nearest neighbor. In order to process $\mathbf{x}$ we compare it with examples in the data set until we have either (1) found k neighbors within the cutoff distance d, in which case we eliminate it as it cannot be an outlier, or (2) we have compared it with all N examples in the data set and failed to find k neighbors within distance d, in which case it is classified as an outlier.

We can think of this problem as a set of independent Bernoulli trials where we keep drawing instances until we have found k successes (k examples within distance d) or we have exhausted the data set. Let $\pi(\mathbf{x})$ be the probability that a randomly drawn example lies within distance d of point x, let Y be a random variable representing the number of trials until we have k successes, and let $P(Y = y)$ be the probability of obtaining the kth success on trial y. The probability $P(Y = y)$ follows a negative binomial dis-

that preserve independence of examples.

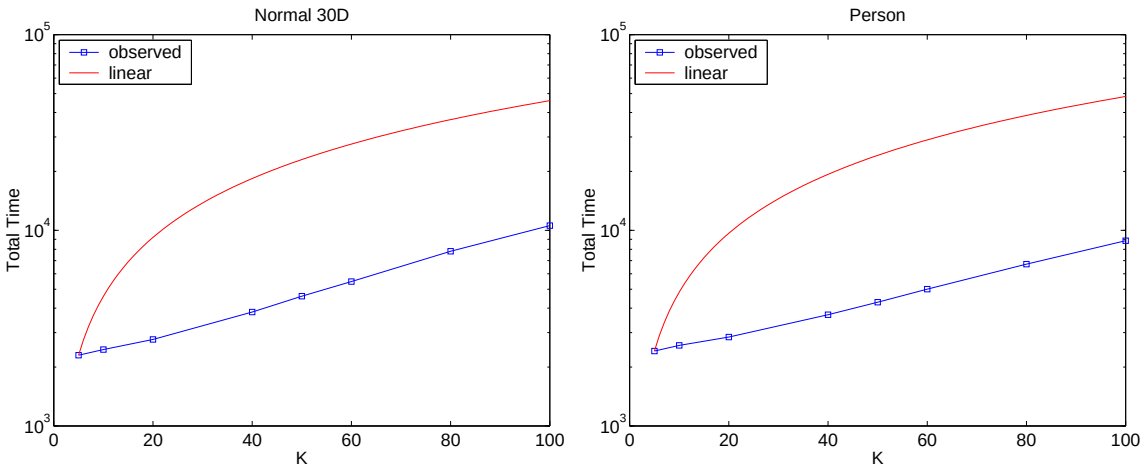

Figure 2: Total time (CPU and I/O) taken to mine outliers as k increases. The top curved line represents the theoretical time taken by an algorithm linear in k based on scaling the observed time for $k = 5$.

tribution:

$$P(Y = y) = \binom{y-1}{k-1} \pi(\mathbf{x})^k (1 - \pi(\mathbf{x}))^{y-k} \quad (1)$$

The number of expected samples we need to draw to process one example x is:

$$E[Y] = \sum_{y=k}^{N} P(Y = y)\ y + \left(1 - \sum_{y=k}^{N} P(Y = y)\right) N \quad (2)$$

The first term is the expectation of concluding a negative binomial series within N trials. That is, as we are processing an example, we keep drawing more examples until we have seen k that are within distance d, at which point we eliminate it because it cannot be an outlier. The second term is the expected cost of failing to conclude the negative binomial series within N trials, in which case we have examined all N data points because the example is an outlier (less than k successes in N trials).

The expectation of a negative binomial series with an infinite number of trials is,

$$\sum_{y=k}^{\infty} \binom{y-1}{k-1} \pi(\mathbf{x})^k (1 - \pi(\mathbf{x}))^{y-k}\ y = \frac{k}{\pi(\mathbf{x})} \quad (3)$$

This is greater than the first term in Equation 2. Combining Equations 2 and 3 yields,

$$E[Y] \leq \frac{k}{\pi(\mathbf{x})} + \left(1 - \sum_{y=k}^{N} P(Y = y)\right) N \quad (4)$$

Surprisingly, the first term which represents the number of distance computations to eliminate non-outliers does not depend on N. The second term, which represents the expected cost of outliers (i.e, we must compare with everything in the database and then conclude that nothing is close) does depend on N, yielding an overall quadratic dependency to process N examples in total. However, note that we typically set the program parameters to return a small and possibly fixed number of outliers. Thus the first term dominates and we obtain near linear performance.

One assumption of this analysis is that the cutoff distance is fixed. In practice, the cutoff distance changes with different values of N. However, we should expect that if the cutoff distance increases with larger N, then scaling will be better as $\pi(\mathbf{x})$ is larger and any randomly selected example is more likely to be a success (neighbor). Conversely, if the cutoff distance decreases, the scaling will be worse. In Figure 3 we plotted the relationship between b, the empirical scaling factor, and c_{50K}/c_{5K}, the ratio of the final cutoffs for $N = 50000$ and $N = 5000$ for the six data sets used in the previous section. We also plotted results for two additional data sets, Uniform 3D and Mixed 3D, which we believed would be respectively extremely difficult and easy. Uniform 3D is a three-dimensional data set generated from a uniform distribution between [-0.5,0.5] on each dimension. Mixed 3D is a mixture of the uniform data set (99%) combined with a Gaussian (1%) centered on the origin with covariance matrix equal to the identity matrix.

The results indicate that for many data sets the cutoff ratio is near or greater than 1. The only data set with an extremely low cutoff ratio was Uniform3D. The graph also indicates that higher values

of the cutoff ratio are associated with better scaling scores (lower b). This supports our theory that the primary factor determining the scaling is how the cutoff changes as N increases.

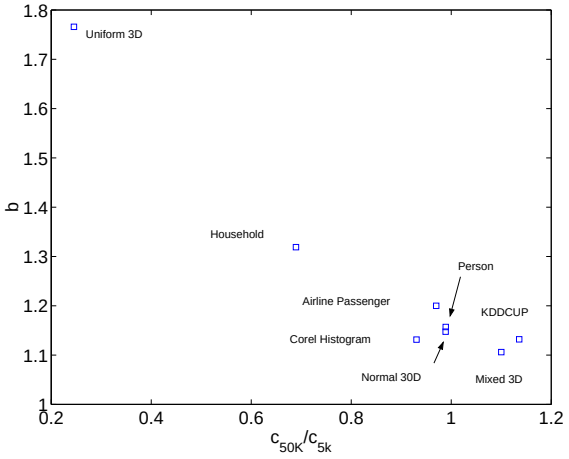

Figure 3: Empirical scaling factor b versus c_{50K}/c_{5K}, the ratio of cutoff scores for $N = 50,000$ and $N = 5,000$.

Figure 4 shows the running time plot for Uniform 3D and Mixed 3D. We expected Uniform 3D to have extremely bad scaling performance because it has no true outliers as the probability density is constant across the entire space. Increasing N simply increases the density of points and drops the cutoff score but does not reveal rare outliers. In contrast, the results for Mixed3D were extremely good ($b = 1.11$). In this data set, as we increase N we find more extreme outliers from the Gaussian distribution and the cutoff distance increases, thus improving pruning efficiency. Finally, we note that data sets with a true uniform distribution are probably rare in real domains.

5 Outliers in Census Data

We have applied our algorithm to mining outliers in the Airline Passenger database. However, for security reasons, we cannot describe the data set nor the discovered outliers in detail. Instead, we present results from the Household and Person data sets to give the readers a qualitative feel for outliers that can be mined from large data sets. We report selected results from running our outlier detection algorithm to return the top 30 outliers with $k = 5$. The full list of top 30 outliers for both household and person are available online[2] and we encourage readers to view this list directly.

[2]http://www.isle.org/∼sbay/papers/siam03/

The top outlier in the household database is a single family living in San Diego with 5 married couples, 5 mothers, and 6 fathers. In the census data, a family is defined as a group of persons related by blood, adoption, or marriage. To be considered a mother or father, the person's child or children must be present in the household. The house had a reported value of $85K and was mortgaged. The total reported income of the household was approximately $86K for the previous year.

Another outlier is a single-family rural farm household in Florence, South Carolina. The house is owned free and clear by a married couple with no children. This example is unusual because the value of the house is greater than $400K (not including the land), and they reported a household income of over $550K.

In the person data set one of the most extreme outliers was a 90+ year old Black Male with Italian ancestry who does not speak English, was enrolled in school[3], has a Doctorate degree, is employed as a baker, reported $110K income of which $40K was from wages, $20K from business, $10K from farm, $15K from welfare, and $20K from investments, has a disability which limits but does not prevent work, was a veteran of the U.S. armed forces, takes public transportation (ferry boat) to work, and immigrated to the U.S. 11-15 years ago but moved into his current dwelling 21-30 years ago. Clearly, there are inconsistencies in this record and we believe that this record represents an improperly completed form.

A second outlier was a 46 year old, White, widowed female living with 9 family members, two of which are her own children. She has a disability that limits but does not prevent her work as a bookkeeper or accounting clerk in the theater and motion picture industry. She takes public transportation to work (bus or trolley) and it takes her longer than 99 minutes to go from home to work.

A third outlier was a 19 year old, White, female with Asian ancestry and Mexican Hispanic origin with a disability that limits but does not prevent work. She earned $123K in business income, and $38K in retirement income (which may include payments for disabilities), and is also enrolled in school.

Finally, we ask readers to keep in mind that these outliers were discovered using a base set of features which were not collected for a specific task (e.g., security screening). In our own work on the airline passenger database, we are spending much effort on feature engineering to get the most relevant information for our algorithm.

[3]Taking a course that a high school or college would accept for credit would count under Census definitions.

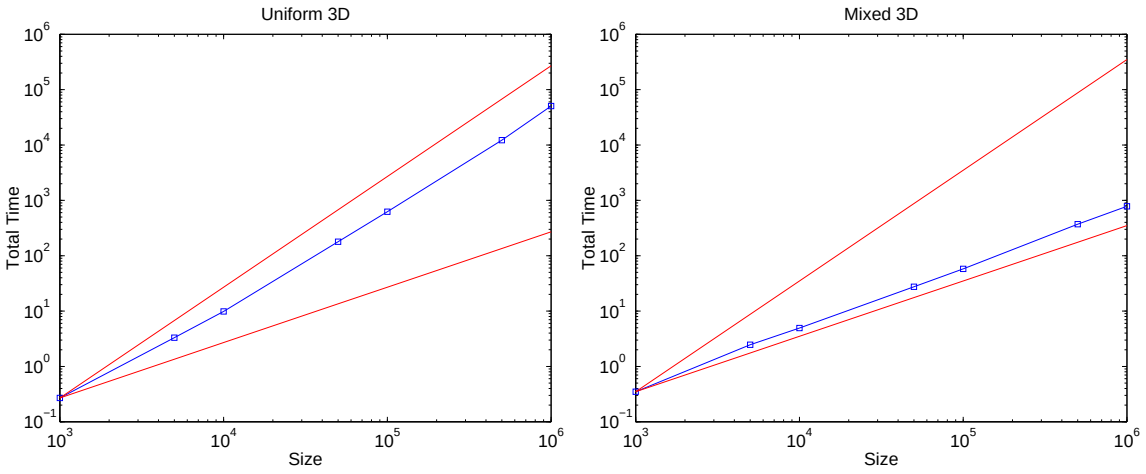

Figure 4: Total time (CPU and I/O) taken to mine outliers on data sets. For Uniform 3D $b = 1.76$, and for Mixed 3D $b = 1.11$.

6 Limitations and Future Work

In this paper, we addressed one roadblock toward applying distance-based outlier detection algorithms to security screening: dealing with the computation time required for large data sets. However, we feel that there are several limitations in applying outlier detection to security databases and that further research needs to be conducted to address these.

The largest and most pressing limitation is that our work has only addressed finding outliers in the data sets that can be represented with a vector space or equivalently a single table in a database. Clearly, almost all real data sources will be in the form of relational databases with multiple tables that relate different types of information about each other.

To address relational data, the simplest solution is to flatten the database with join operators to form a single table. While this is a convenient solution it loses much of the information available. For instance, a flattened database cannot easily represent passengers that have a variable number of flight trips. We also found that flattening a database could create dependencies between examples and this can reduce the effectiveness of randomization and pruning.

We are currently investigating how we can extend our algorithm to handle relational data natively. There are two research questions that arise. First, how does one define a distance metric to compare objects which may have a variable number of linked objects? There has been some work on defining metrics to work on relational data [6, 9, 16]. The central idea is to apply a recursive distance measure. That is, to compare two objects one starts by comparing their features directly, and then moves on to compare linked objects and so on. Second, how does one efficiently retrieve an object and its related objects to compare them in the context of searching for outliers? Retrieving related objects may involve extracting records in a non-sequential ordering and this can greatly slow database access.

A second area we did not address in this paper is determining how to set algorithm parameters such as k, the distance measure, and the score function. Each of these parameters can have a large effect on the discovered outliers. In supervised classification tasks one can set these parameters to maximize predictive performance by using a hold out set or cross-validation to estimate out-of-sample performance. However, outlier detection is unsupervised and no such training signal exists.

Finally, our approach combines randomization and pruning to speed the computation of the top outliers in the data set. Our method gives exactly the same results as computing all N^2 pairwise distances between examples and from these distances selecting the most extreme outliers. Alternatively, one could use a small random subset of examples to determine the outliers in the entire data set. This would require $O(NN_s)$ distance computations where N_s is the size of the subset. This subset approach is not guaranteed to return the same outliers as performing all N^2 pairwise comparisons but it may perform adequately. Our initial experiments indicate that the correspondence with the full N^2 comparison strongly depends on the data set. For example, on Mixed3D a subset of 1000 examples was sufficient to give 90% correspondence whereas on the Person data set a subset of 1000 points gave less than 30% correspondence and 1,000,000 examples resulted only in 70% correspon-

dence. We plan to investigate this issue further.

7 Conclusions

In our work applying outlier detection algorithms to large, real databases a major limitation has been scaling the algorithms to handle the volume of data. In this paper, we presented an algorithm based on randomization and pruning which finds outliers on many real data sets in near linear time. This efficient scaling allowed us to mine data sets with millions of examples and many features.

Acknowledgments

We thank Thomas Hinke and David Roland of NASA Ames for their help with the airline passenger data and for reviewing a draft of this paper. This work was supported by the CICT Program at NASA Ames Research Center under grant NCC 2-5496.

References

[1] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 2001.

[2] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *Proc. of the Sixth European Conf. on the Principles of Data Mining and Knowledge Discovery*, pages 15–26, 2002.

[3] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 1994.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] S. Berchtold, D. Keim, and H-P Kreigel. The X-tree: an index structure for high-dimensional data. In *Proc. of the 22nd Int. Conf. on Very Large Databases*, pages 28–39, 1996.

[6] G. Bisson. Learning in FOL with a similarity measure. In *Proc. of the Tenth National Conf. on Artificial Intelligence*, pages 82–87, 1992.

[7] M. M. Breunig, H.P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 2000.

[8] Commission on Engineering and Technical Systems. *Assessment of Technologies Deployed to Improve Aviation Security: First Report*. Number NMAB-482-5. National Academy Press, 1999.

[9] W. Emde and D. Wettschereck. Relational instance-based learning. In *Proc. of the thirteenth Int. Conf. on Machine Learning*, 1996.

[10] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Data Mining for Security Applications*, 2002.

[11] Federal Aviation Administration. Security of checked baggage on flights within the united states – proposed rule. *Federal Register*, 64(74):19219–19240, April 19, 1999.

[12] E. Fix and J. L. Hodges. Discriminatory analysis: Nonparametric discrimination: Small sample performance. Technical Report Project 21-49-004, Report Number 11, USAF School of Aviation Medicine, Randolf Field, Texas, 1952.

[13] R. Guttmann. A dynamic index structure for spatial searching. In *Proc. of the 1984 ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.

[14] D. Hawkins. *Identification of outliers*. Chapman and Hall, 1980.

[15] S. Hettich and S. D. Bay. The UCI KDD archive. [http://kdd.ics.uci.edu/]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

[16] T. Horvath, S. Wrobel, and U. Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43:53–80, 2001.

[17] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proc. of the 25th VLDB Conf.*, 1999.

[18] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *VLDB Journal: Very Large Databases*, 8(3-4):237–253, 2000.

[19] T. Lane and C. Brodley. An application of machine learning to anomaly detection. In *Twentieth Annual National Information Systems Security Conf.*, volume 1, pages 366–380, 1997.

[20] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, 1999.

[21] W. Lee, S. Stolfo, and K. Mok. Adaptive intrusion detection: a data mining approach. *Artificial Intelligence Review*, 14(6):533–567, 2000.

[22] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proc. of ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[23] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. of the ACM SIGMOD Conf.*, 2000.

[24] S. Ruggles and M. Sobek. Integrated public use microdata series: Version 2.0. [http://www.ipums.umn.edu/], 1997.

[25] U.S. House of Representatives, Subcommittee on Aviation. Hearing on aviation security with a focus on passenger profiling. http://www.house.gov/transportation/aviation/02-27-02/02-27-02memo.html, February 27, 2002.

The Link Discovery Problem and Characteristics of Real-World Databases

Jafar Adibi
Information Sciences Institute
University of Southern California
4676 AdmiraltyWay, Marina del Rey, CA 90292
adibi@isi.edu

Abstract

An implicit assumption in some of the studies in statistics, machine learning and data mining is that the data is uniformly distributed or exhibits either Poisson or binomial distribution. However, most of the databases in real world applications violate these assumptions and disobey such distribution. Link Discovery (LD) as a new challenge in data mining is not an exception in this regard. In this paper we briefly describe the LD challenge problem and we show that these datasets are also skewed and exhibit Zipf law distribution. Since issues such as privacy and security promote the use of simulated database in data mining in general and specifically in LD, the need to use of simulated data for performance evaluation of these techniques is inevitable. In this study we provide some major characteristics of a real world database and argue that any simulated database has to follow these characteristics to justify the performance analysis and evaluation of LD techniques.

Keywords

Relational Data Mining, Link Discovery, Zipf Law

1. Introduction

During the past year, the development of information technology that could aid organizations in their efforts to detect and prevent fraudulent activities has become an important topic for research and development. Scince the amount of information, tips, reports and data increases exponentially every day, analyzing such data manually to find patterns and to identify suspicious activities is impossible. Hence, new techniques to achieve these goals and to overcome such obstacles are needed.

The Evidence Extraction and Link Discovery (EELD) program of the Defense Advanced Research Projects Agency (DARPA) is one of the research projects, which attempt to address this issue. The main goal of the EELD program is the development of techniques for mining large amounts of data to find hidden patterns, extract valuable knowledge and discover hidden links among sparse evidences. Such technologies are capable to extract relevant data and relationships about people, places, events, organizations, objects and activities from different sources such as in-house databases, message traffic and open source data. Eventually, these techniques will link items relating potential groups; learn patterns of different scenarios, identify organizations, learn fraudulent behavior, model groups' activities and detect emerging threats.

The EELD program divided its focus into three related sub-areas to achieve its goals. These areas are Evidence Extraction (EE), Link Discovery (LD) and Pattern Learning (PL). While the main focus of the EE area is the task of obtaining structured evidence data from unstructured sources, LD's main goal is the task of identifying complex, multi-relational patterns that indicate potentially threatening activities in large amounts of structured data. PL's main concern is the automated discovery of new relational patterns.

In this paper we report part of the issues related to an on going project at USC information Sciences Institute called KOJAK[1]. KOJAK is hybrid link discovery system developed under the EELD program. It combines knowledge representation and reasoning technology with statistical clustering and analysis techniques from the area of data mining.

Similar to other machine learning and data mining techniques practically LD needs to evaluate the developed algorithm on some databases. Whether one builds a new algorithm, or wishes to use an existing one, it is important to know how the algorithms perform. Since privacy and security are important concerns in these domains the use of synthetic data for evaluation, modification and tuning of new technologies is inevitable. A true advantage of synthetically generated datasets is that they are highly tunable as to the level of noise or corruption present in the evidence.

[1] http://www.isi.edu/~hans/eeld/

Table 1: Comparison between Relational Models and Models of Relational Data (RDM) (Adapted from [Senator, 2001])

	Relational Models	Models of Relational Data
Data	One type of entity with a set of attributes	Non-homogeneous, multi-source, uncertain data
Data Type	Homogeneous data	Many types of entities, each with own set of attributes
Attributes	Assume independence among entities	Entities are interdependent
Representation	Nodes: variable Links: statistical relationship between variables	Nodes: person, place, organization, event, account… Links: relation between two objects (nodes)
Focus	Probability that a given node in the data can be assigned a variable, given other variables, based on probabilities derived from statistical analysis of these variables	Likelihood that an instance of a specific graph-theoretic structure in the data matches a pattern of interest, include temporal, spatial, organizational, and/or transactional patterns.

This paper attempts to study the issue of evaluation through synthetic data rather than real world databases in LD applications. We study and analyze an example of a real world databases (a.k.a. Russian Contract Killing) and discuss the effect of such characteristics on design of synthetic databases. Our main finding is that these datasets are skewed and exhibit *Zipf* law distribution and do not follow the conventional assumption of binomial and /or Poisson distribution.

The rest of this paper is organized as follows.We start with a brief description of the LD problem, which includes a short review on Relational Data Mining along with its differences to other areas of data mining. In Section 3 we review characteristics of some other real world databases along with *Zipf* law distribution, its characteristics and presence in the real world. In Section 4 we introduce a real-world and a synthetic LD database, which have been studied in this paper. In section 5 we report our finding on studying of such databases followed by concluding remarks.

2. Link Discovery

In this section we explain the problem of LD in more detail. At first we provide a short review of the area of Relational Data Mining. Next, we define the problem of LD and its main differences to other areas of DM.

Relational Data Mining

Conventional Knowledge Discovery in Databases (KDD) techniques assume that the data to be mined is in a single relational table and that examples are flat tuples of attribute values [Piatetsky-Shapiro, 1996][Fayyad, 1996]. The attribute-value paradigm only allows the analysis of simple objects. It requires that each object can be described by a fixed set of attributes each of which can only have a single value. The major focus of machine learning techniques, KDD and statistics has been on classification, clustering and regression using feature vectors as inputs. Properties of typical data mining tasks and attention to scaling, efficiency and the dimensionality curse are the reason why mining large relational databases containing more than one table has been given less attention in the past.

New areas of data mining such as entity mining and link discovery concerns mining data from multiple relational tables that are richly connected. To be able to represent more complex and structured objects, one has to employ a relational database containing multiple tables. Each object can be described by multiple records in multiple tables. To be able to analyze relational databases containing multiple relations properly, specific algorithms will have to be written that cope with the structural information that occurs in relational databases. Mining this type of data is referred to Relational Data Mining (RDM) [Dzeroski, 2001][Koller, 1998]. RDM aims to integrate results from existing fields such as inductive logic programming, artificial intelligence, data mining, machine learning and relational databases[Mooney, 2002].

In RDM, different form other data mining techniques, the knowledge to be discovered may be the definition of another relation rather than a classification, clustering or regression function. Mining databases that consist of complex structured objects also falls within the scope of this field: the normalized representation of such objects in a relational database requires multiple tables.

Present RDM approaches consider all of the main data mining tasks, including association rules analysis, classification, clustering, learning probabilistic models and regression. The techniques used by single-table data mining approaches for these tasks have been extended to the multiple-table case including relational association rules, relational classification, relational decision trees, and probabilistic relational models, etc. RDM methods have been successfully applied across many application areas, such as analysis of business data, bioinformatics, pharmacology and Web mining [Dzeroski, 2001]. Table 1 (adapted form [Senator, 2001]) illustrates the major difference between Relational Models (conventional KDD) and Relational Data Mining (RDM).

Link Discovery Functionality

LD is a crucial step in data mining to counter-fraudulent activities and is concerned with the identification of complex relational patterns that indicate potentially threatening activities in large amounts of relational data. The main goal of LD is to identify related objects, entities,

Link	Object	Attribute
Surname	John_Smith	Smith
FirstName	John_Smith	John
HasGender	John_Smith	Male
Surname	Michael_Green	Green
FirstName	Michael_Green	Michael
HasGender	Michael_Green	Male
CITIZENSHIP	John_Smith	Russia
Surname	Joe_Brown	Brown
FirstName	Joe_Brown	Joe
HasGender	Joe_Brown	Male
Killer	Action_3	John_Smith
Killer	Action_3	Michael_Green
Victim	Action_4	John_Smith
Killer	Action_4	Joe_Brown
Linked	John_Smith	Location_2
Found_at_Scene	John_Smith	Location_4
Member	John_Smith	Org_34
Employee	Joe_Brown	Org_1
Killed_by	Mary_Jones	John_Smith
Associate	John_Smith	Michael_Green
Killed_by	John_Smith	Joe_Brown
Used	John_Smith	Weapon_2
Used	Michael_Green	Weapon_2
Murder_Weapon	John_Smith	Weapon_1
Murder_Weapon	Joe_Brown	Weapon_1
Used	John_Smith	Weapon_25
Has-Occupation	John_Smith	Criminal
Has-Occupation	Michael_Green	Criminal
Has-Occupation	Joe_Brown	Criminal

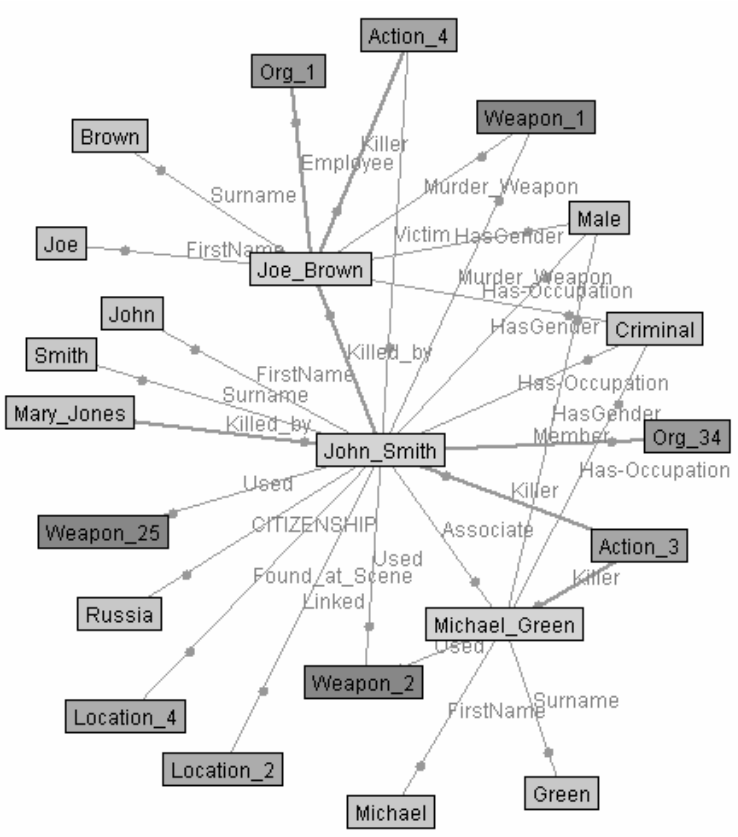

Figure 1: A simple example of the Link Discovery problem adapted form the Russian Contract Killing (RCK) domain. *Left*: relations among objects presented in a table. *Right*: illustration of such relations to demonstrate the complexity of the domain.

events, persons, organizations and plans which lead to identifying known, complex and multi-relational patterns that indicate potentially threatening activities[1] . The data and patterns include representations of people, organizations, objects, actions and events and many types of relations between them including the time tag, which indicated the time of occurrence of an event. Given the style of data needed for LD, it requires new methods in the area of relational data mining.

Figure 1 shows a simple example of the link discovery problem. The left side of figure shows the relations among objects in the database. For simplicity, we illustrate all relations in one table. The first column indicates the relation, the second column is the object and the third column indicates the attribute of the relation. Right side of Figure 1 is an illustration of such relations as links among objects and attributes.

Some issues that make LD a rather challenging task include the following:

- **Connectivity**: many real world LD databases suffer form connectivity curse. To be more specific, almost all objects in the database including people, organizations, locations etc. are connected directly or indirectly. Hence, there is a high probability that two objects are connected to each other through these attributes. For instance in our example in Figure 1 *Joe-Brown* and *John-Smith* are connected among other things because both *genders* are *male*. The *male* attribute might not be an important relation, but if both *Joe-Brown* and *John-Smith* share the same nationality that may raise a flag for an analyst.
- **Negative Noise (Observablity):** This factor describes how much of the real data have been observable to an analyst. Not all events, that occure in the world state, will be known to LD components. Many incidents may not be reported or might not be observable. For instance, the relationship of two people as friend or teammate

[1] http://www.darpa.mil/iao/EELD.htm

may not reported in data while it might be a very important factor.

- **Positive Noise**: There are some evidences that are the result of tasks that are very similar to a suspicious behavior or show a strong connection among parties but they are not. These patterns are similar in certain ways to suspicious patterns and they are not easy to differentiate from others.
- **Corruption**: Corruption are varies from typo and misspelling in names and numbers, to assigning a wrong value to a given attribute. An example would be naming the wrong hitman as the murderer in a contract killing case.
- **Complexity**: On one hand, the richer the information available to a LD system, the more interesting relationships between entities can be discovered. On the other hand, many interesting connections will not be findable unless appropriately rich background knowledge is available in whose context the relations extracted by EE methods can be analyzed.

LD techniques aim to overcome these obstacles to discover hidden knowledge in large databases. Examples of the hidden knowledge are:

- Association between individual X and organization Y;
- Link among facility F, owned by Y, and individual X.
- Similarity between agent A's and agent B's behaviors.
- Membership of agent A and agent B to the same organization
- Strong connection between individual X and individual Z.

Link Discovery and Other Areas of Data Mining

LD is different from other areas of data mining based in its view, techniques, focus and methodology. For instance, LD techniques are not looking for similarities among individuals as a classification task. There can be a strong connection, relation or link between two individuals while there are no similarities among them. While clustering and classification techniques try to maximize the distance among classes and minimize the distance within classes LD's main focus is to find strong, valuable and informative links among individuals or classes.

Moreover, LD has a fundamental difference with outlier detection. Outlier detection tries to solve the problem of detecting rare events, deviant objects, and exceptions. In addition, usually there is a need for a database including normal cases, abnormal cases along with positive and negative noise. However, in real-world LD applications there is a strong need to analyze and detect a threat by mining a series of events even with only a few or no previous similar cases.

Another aspect of the LD problem is adaptation. Individuals adapt themselves to the current monitoring policy and data mining techniques frequently. An intelligent LD system has to be capable to handle this issue as well.

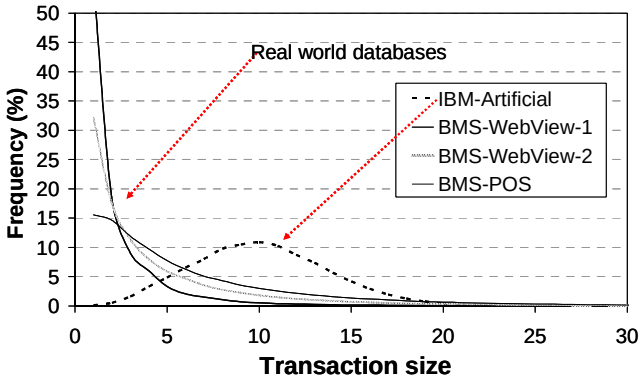

Figure 2: Dataset transaction size distribution. Adapted from [Zheng, 2001]

3. Real World Database

Before describing the LD database in Section 4 we take a quick look on some other real-word datasets and outline new findings about the distribution of such data. Recognized groups of data typically are skewed and exhibit fractal dimension. Many physical systems contain a form of functional self-similarity that owes its richness to recursion [Schroeder, 1991]. For instance, most biological systems contain self-similar structures that are made through recurrent processes [Mandelbrot, 1982]. Human brains, ocean flows, changes in the yearly flood levels of rivers, voltages across nerve membranes, musical melodies and economic markets also create enormously complex behavior that is much richer than the behavior of the individual component units. New findings in different branches of science and technology also show the presence of self-similarity and fractal dimensions in different domains [Mandelbrot, 1982][Mandelbrot, 1965]. To name a few: medical diagnosis (physician treatment and patient response), robot navigation (robot move and environment response to robot sensors), Internet web logs and network behavior monitoring (packet transmission, switch behavior and network response), recent measurements of local-area and wide-area traffic of traffic data from networks and services such as ISDN traffic, Ethernet LAN's, Common Channel Signaling Network (CCNS) and Variable Bit Rate (VBR) video are examples of such phenomenon. Many of these data exhibits variability at a wide range of time scales. [Willinger, 1997][Adibi, 2001][Burlaga, 1986][Faloutsos, 2001][Feder, 1988][Hsu, 1990]

These observations raise caution on designing synthetic data. For evaluation of a data mining technique synthetic data has to be quite similar to real world data, otherwise the evaluation, performance measurement and the result provided on synthetic data might be totally different when such techniques are applied to real world databases. The

difference between real world data and simulated data has been explored in related work outlined below.

Zijian Zheng et al, in [Zheng, 2001] have studied the difference between synthetic data and real world database in market basket analysis through association rules or frequent itemset. In their work, they compared five well-known association rule algorithms using three real-world datasets and an artificial dataset from IBM Almaden. Not so surprisingly, they have shown that the experimental results confirmed the performance improvements previously claimed by the authors on the artificial data, but some of these gains do not carry over to the real datasets. Fig 2. adapted from [Zheng, 2001] shows the difference between a real world and an artificial data set provided by IBM Almaden. Fig. 2 shows the frequency relevant to transaction size (items bought by a given customer). As it shows in the Fig. 2 there are many transaction sizes with low frequency and a few transaction sizes with high frequency. Such distribution fits in Zipf law distribution.

Our own observation also shows that the distribution of a real world database follows the Zipf law distribution. In the following we first briefly review the Zipf law distribution and its characteristics followed by description of a real world LD database exhibiting it.

Zipf Law

Zipf's law usually refers to the size of an occurrence of an event relative to its rank. The two Zipf laws are: the *rank-frequency* and the *frequency-count*. Let $f(r)$ be the occurrence frequency of the r^{th} most frequent items in a given set.[Hill, 1974][Adamic, 1974]

The *rank-frequency* plot is the plot of the occurrence *frequency* $f(r)$ versus the *rank r*, in log–log scales. The *rank-frequency* version of Zipf's law states that $f(r) \propto 1/r$. This is typically referred to as the Zipf 's law or the Zipf distribution. In log-log scales, the Zipf distribution gives a straight line with slope -1. The generalized Zipf distribution (Zipf-like) is defined as $f(r) \propto 1/r^{\theta}$ where the log-log plot can be linear with any slope. The second law, also known as the discrete Pareto distribution, involves the *count-frequency* plot: let $c(f)$ be the count of items that appear *f* times in the document. The second Zipf's law states that $c(f) = 1/f^{\varphi}$.[Zhiqiang, 2001]

The *count-frequency* plot actually corresponds to the probability density function of the occurrence frequency of an item in a set and it is a mathematical consequence of the first law. Also, in log-log scales, the count-frequency plot of a Zipf distribution will be a straight line, with slope φ. This graph is illustrated in Fig. 3.

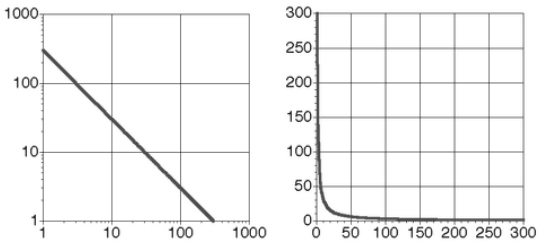

Figure 3: Illustration of Zipf distribution

Zipf's law indicates that in such a distribution a few elements score very high (the right tail in Fig. 3). Examples of such fact are: [Faloutsos, 2001]

- Language: words that are used extremely often
- Library: books that everybody wants to borrow
- Marketing: products that every one wants to buy
- Websites: websites or pages with lots of interest
- Olympic: counties with many number of medals

In addition it indicates that at the end a large number of elements score very low (the left tail in Fig. 3). Examples of such fact are:

- Language: words that are almost never used
- Library: books that are almost never checked out
- Marketing: many unpopular products
- Website: sites or pages with almost no hits
- Olympic games: countries that never gets a medal

Other examples are: distribution of file sizes (Zipfs law), income distribution (Pareto's law), publication counts (Lotka's law), length of articles in a newspaper (Zipf), web hit counts (Huberman) duration of UNIX jobs (Harchol-Balter) and length of file transfers (Bestavros+). [Faloutsos, 2001]

4. Link Discovery Databases

Due to respecting people privacy, access to real world databases has been a main concern in KDD community in past. The LD community is not an exception in this matter. In particular, since the LD goal is to relate people, place and entities, it triggers concern with privacy issues. The balance between privacy concerns and the need to explore large volumes of data for LD is difficult problem. The development of technology for KDD and LD has revitalized concern about the following general privacy issues: secondary use of personal information, handling misinformation, and granulated access to personal information. These issues motivate simulated data for performance evaluation of LD techniques. In this section we explain the characteristics of a real-world and an artificial dataset and we compare some of their characteristics.

The Russian Contract Killing (RCK) Data

The dataset of contract killings was first compiled by O'Hayon and Cook [Cook, 2000] through Veridian Systems Division (VSD) [Williams, 2002]. This effort was a response to research on Russian organized crime that encountered frequent. Each of the contract-killing reports provided an image of the criminal scene in Russia, but there was no information of how these were linked, what the trends were, who the victims were, the relationship between victims themselves or the relationship between victims and perpetrators.

Table 2: Russian Contract Killing Database Characteristics, Links and Statistics

Entities	#	Events	Persons	Organizations	Locations	Materials	Weapons
Events	169	59	--	--	--	--	--
Persons	615	531	1,071	--	--	--	--
Organizations	320	304	472	140	--	--	--
Locations	206	n/a	90	33	2	--	--
Materials	7	6	7	2	4	0	--
Weapons	148	128	268	11	19	2	1
Sources	265	265	n/a	n/a	n/a	n/a	n/a
Occupations	n/a	n/a	532	n/a	n/a	n/a	n/a

The database was captured as a series of incidents. Each incident in the series received a description of the information drawn from the sources, typically one news article, but occasionally more than one.

Information in the series is based on open source reporting, especially Foreign Broadcast Information Service and Joint Publications Research Service. In addition additional materials on the World Wide Web were consulted. Most of the reported killings are from the 1990s. The main focus is on Russia, but killings involving Russian organized crime outside Russia are also included.

In total 166 series incidents have been coded. Database includes 1,630 entities including the source entities and 3,415 links not including additional 532 Person-Occupation relations. The database has over 40 relational tables. The number of tuples in a relational table varies from 800 to as little as 2 or 3 elements. The data is presented as a series of events in a relational database format. This format contains objects described in rows in tables, each of which has attributes of differing types. The objects consist of the following:

- Entity Objects: *Location*, *Material*, *Organization*, *Person*, *Resource*, and *Weapon*.
- Event Objects: generic *Event*
- Links: used for expressing links between/among Entities and Events, and currently consisting of those represented by gray color in Table 2.

Synthetic Version o RCK

Synthetic version of the Russian Contract Killing data was generated by two different simulators. One is a Bayesian Network (BN) Simulator [Team, 2002]and the second one is a Task-Based (TB) Simulator [Team, 2002]. Our main focus is on TB simulator.

The Task-Based simulator [Team, 2002] provides a flexible mechanism for creating synthetic datasets within the EELD program. It includes a pattern specification language, a knowledge base, case generation and representation, evidence generation and corruption, and answer key representations. All the core of the task-based simulator are tasks. Each task contains one or more methods, where each method has a probability of being selected given that its preconditions are satisfied. The simulator also provides powerful functionality for filtering and corrupting data. This is particularly important to represent situations where actual data is expected to have low observability. As shown next, the data has been presented in simulation output as binary predicates:

```
(isa UID342 report)
(eventOccursAt UID342 1/2/2002)
(reportSource UID342 Bank)
(reportContent UID342
 (isa UID15622 Withdrawal))
(reportContent UID342
 (transactionAmount
        UID15622 22000))
(reportContent UID342
 (eventOccursAt UID15622
       Jan_1,_2002_2:00:00_AM))
(reportContent UID342
 (transactionAccount
       UID15622 UID10420))
```

The simulation data include meta-data, such as when and where a specific event was reported. The simulator has the capability to generate different numbers of agents, organization, crimes, group activities, etc. with different signal to noise ratio.

For comparison with the real database we report our result on two different synthetic datasets with different characteristics. These datasets are described in Table 3. Test 1 includes about 400 people in its data and Test 2 has about 2000 people entities.

Table 3: Synthetic data selected for the experiments

File Name	Noise	Observability	Connectivity	Size
Test 1	High	Very low	Medium	Medium
Test 2	Medium	Very low	High	Medium

5. Database Analysis

In the course of the analysis of these datasets we were able to study some of the local properties of the LD graph. In this section we survey these observations and point out that traditional random graph models distribution (i.e Normal Distribution) would do a poor job of explaining them. We interpret the whole database as a graph similar to the graph on right side of the Fig. 1. In this graph objects (i.e person, location, action etc.) are connected to each other through their attributes (i.e. male, John, Smith).

Notation

In the following we describe our notation for the rest of this section along with assumptions and definitions. A graph is characterized by the following:

- Nodes that represent entities such as person, place etc.
- Links, connect two nodes to each other such as *Gender*, which connects *Person_124* to Male.
- E, the set of all edges or links (relations) existing in the graph.
- V, the set of all possible nodes in the graph
- $d(i)$, the degree of a node i which is the number of in-links and out-links connected to the node.

Degree Distribution

We begin with the degree of nodes in the LD graph. Fig. 4 is a log-log plot with the x-axis as the number of links connected to each person and y-axis as number of persons with such number of links. The plot suggests that the probability that a person has degree $d(i)$ is proportional to $1/(i)^{\alpha}$ where α is approximately 2. Such Zipf law distributions cannot arise in models that exhibit either a Poisson or a binomial distribution. There is much information available for a few individuals and not very much information for all the others in the database. The total number of links was 6174 links. The average number of links per person was 131.36 and the variance of such distribution was 137.7. In this graph we have considered the following attributes as *links*, which are connected to a given person:

"attacked_by","member", "killed_by",
"owner" "found_at_scene", "citizenship",
"linked", "used","inhabitant","associate",
"employee", and "murder_weapon".

Relation Distribution

In this experiment we count the frequency of "relation-attribute" pairs (i.e. HASGENDER -> MALE) in the database. Interestingly such distribution also shows an exponential distribution. Fig. 5 is a distribution with the x-axis as number of "relation-attribute" pairs and y-axis as "relation-attribute" pairs in the Test 1 database. Some of the most frequent "relation-attribute" pairs among all relations have shown in Table 4.

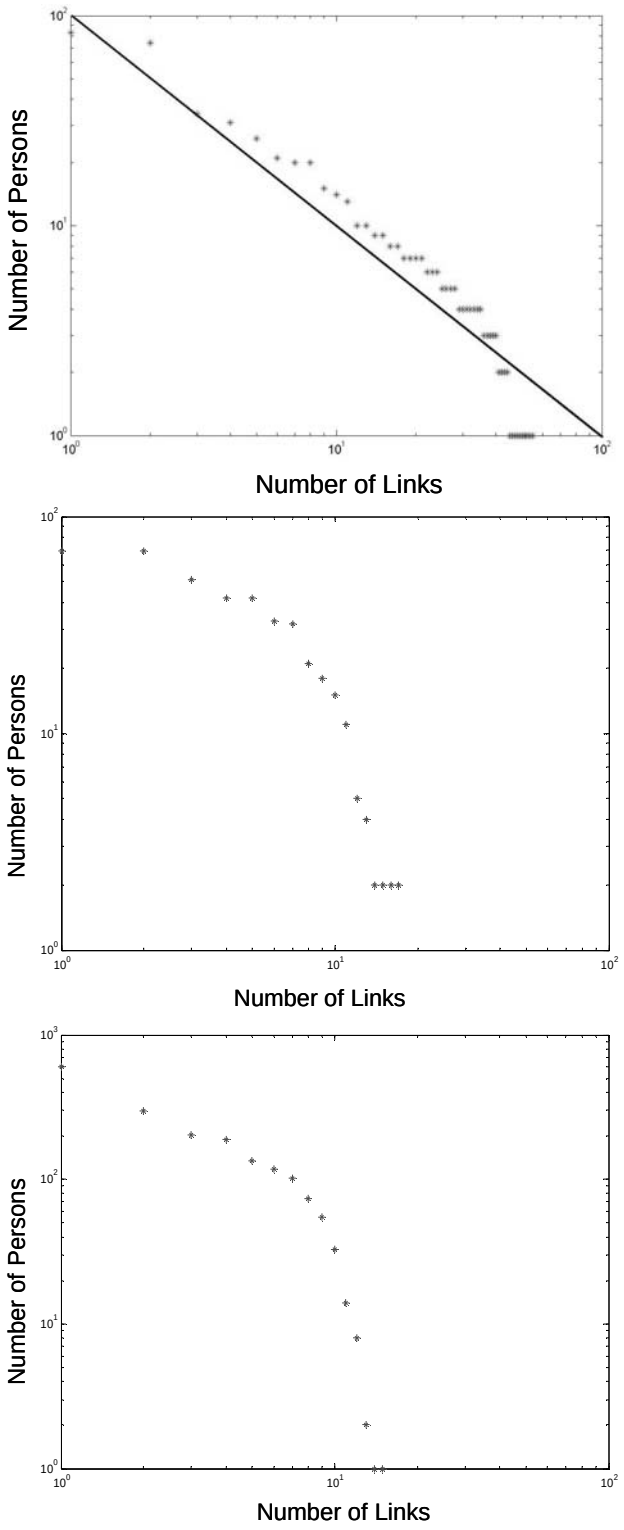

Figure 4: Frequency rank graph illustrating the number of people vs. the number of in and out links in a log-log scale. Up: data belongs to real world database and suggests the Zipf law distribution. The Middle and lower graphs belong to synthetic data Test 1 and Test 2 respectively, which does not follow the Zipf law distribution.

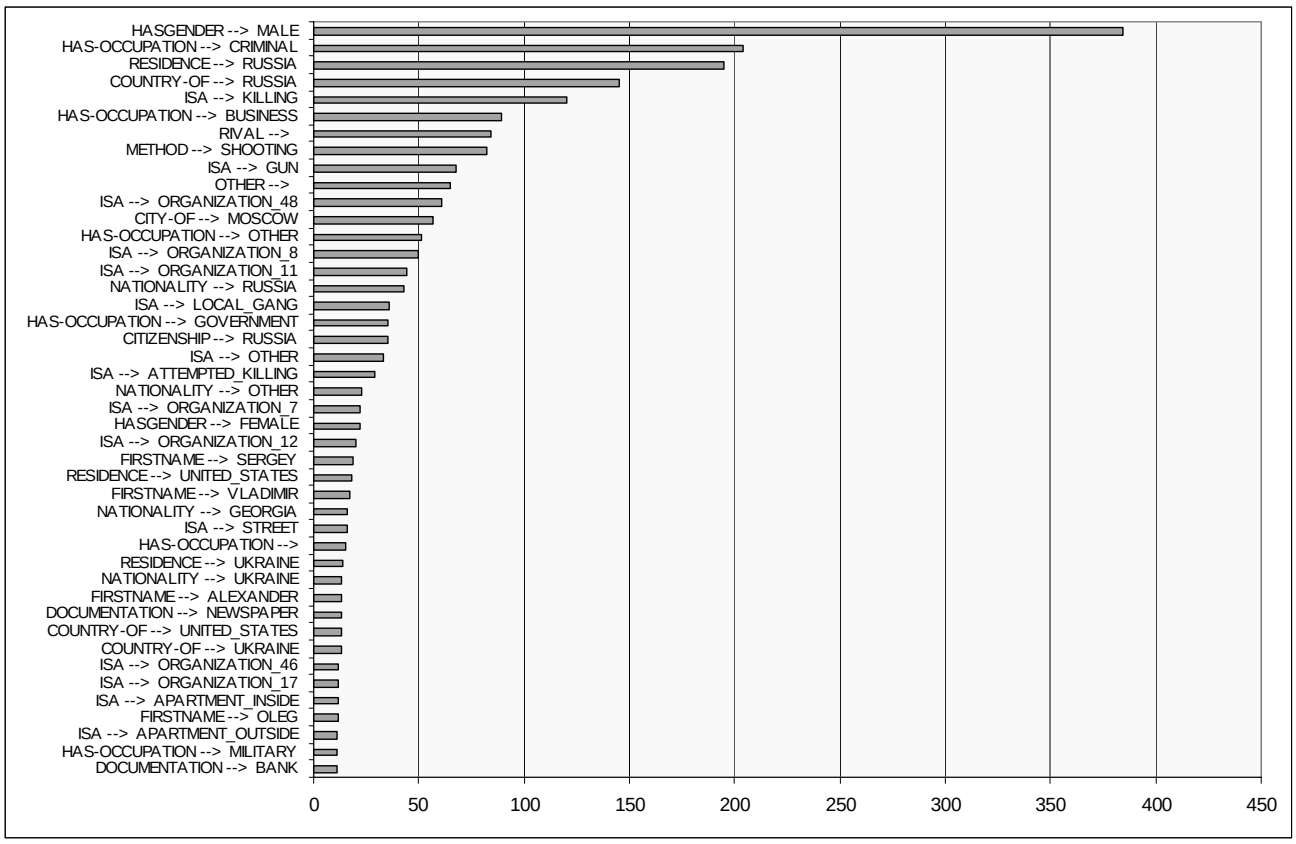

Figure 5: Distribution of number of relations-attribute in RCK database.

Table 4: Most frequent Relation-Attribute in RCK database.

Relation	Attribute	Freq.
Has-Gender	Male	384
Has-Occupation	Criminal	204
Residence	Russia	195
Country Of	Russia	145
Isa	Killing	120
Has-Occupation	Business	89
Methods	Shooting	82
Isa	Gun	68
Isa	Organization_48	61
City Of	Moscow	57
Has-Occupation	Other	51
Isa	Organization 8	50
Isa	Organization 11	44

Fig. 6 is a log-log plot with the x-axis as number of relations-attribute. The distribution looks quiet faintly Zipf distribution and suggests that some of the pairs are very frequent such as "`HASGENDER -> MALE`" and many of them are very unpopular.

Activity Distribution

The second property of the RCK database we were interested in, was the number of murders with respect to individuals in datasets. In this case we were only interested in counting *attacked_by* and *killed_by* links. This means we count the number of people who got killed (or attacked) by a given person. Fig. 7 is a log-log plot with the x-axis as number of links connected to each person. Again the distribution looks faintly like a *Zipf* distribution and suggests that the probability that a person has degree $d(i)$ (*attacked_by* and *killed_by*) to attempt a murder is proportional to the $1/(i)^{\alpha}$, although here the variations seem larger

6. Conclusion

In this paper we studied some characteristics of real world databases for the problem of Link Discovery. While the implicit assumption in many data mining algorithm is that the data is uniformly distributed, many databases in real world applications violate this assumption. They are skewed and exhibit fractal Zipf law distribution and /or some other non-conventional distribution.

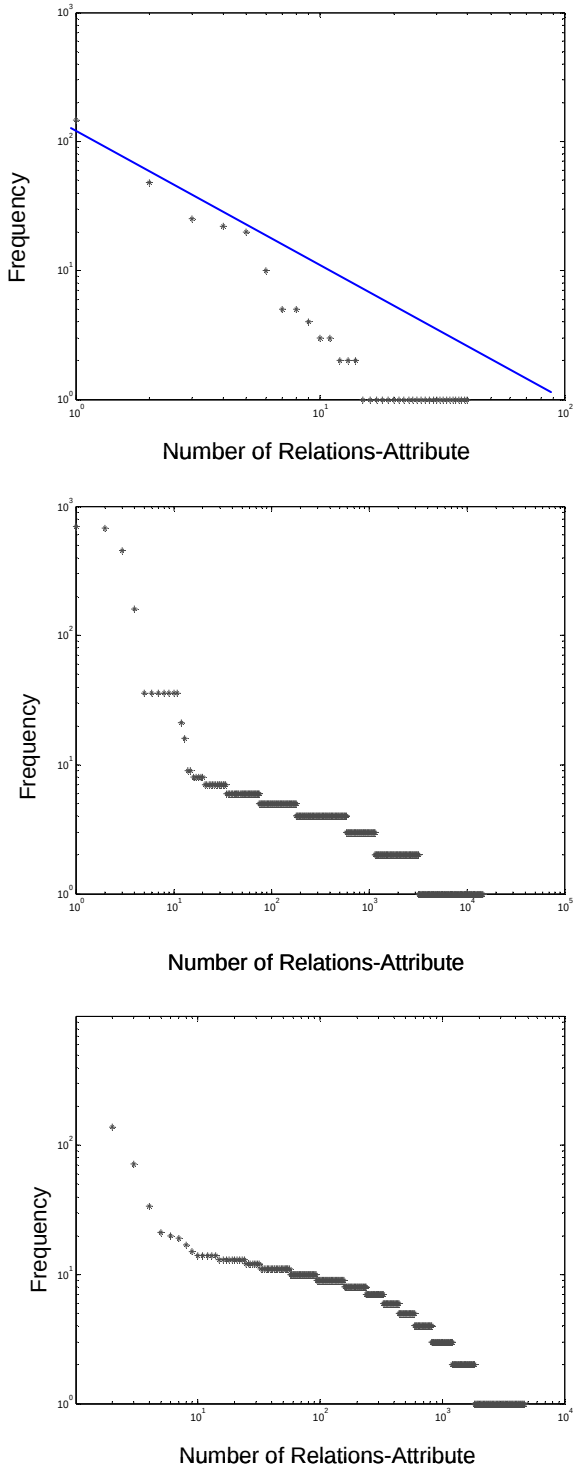

Figure 6: Frequency rank graph illustrating the number of "relation-attribute" pair vs. the frequency of such item in a log-log scale. Up: data belongs to real world database and suggests the Zipf law distribution. The Middle and lower graphs belong to synthetic data Test 1 and Test 2 respectively, which does not follow the Zipf law distribution.

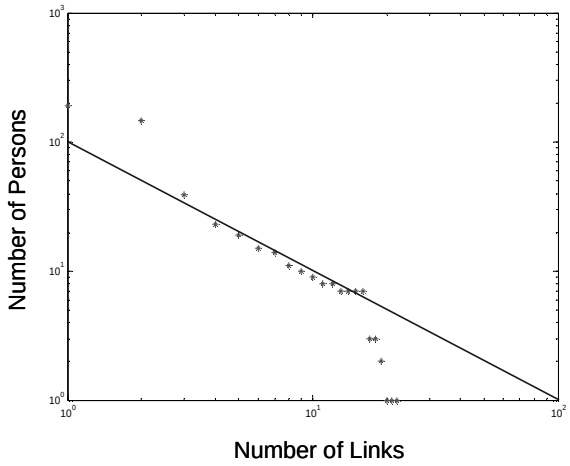

Figure 7: Frequency rank graph illustrating the number of "relation-attribute" pair (only *killed and killed-by)* vs. the frequency of such item in a log-log scale. Data is from Test 1 dataset and suggests the Zipf law distribution.

Synthetic data are considered an important alternative to real world databases. The following are the main factors for such consideration:

- Privacy and security issues increase the need for synthetic data for performance evaluation, algorithm modification and tuning.
- A simulation engine provides powerful functionality for filtering and corrupting data especially where actual data is expected to have low observability.

For synthetic data to be useful it has to adequately represent societal and individual attitudes and it has to be similar enough to the real world data. To be more specific, data distribution for variables has to be similar. As it shown in some of the related work, those techniques which perform successfully on simulated data may fail on real-world datasets. A wide variety of knowledge discovery and LD steps such as data sampling, data cleaning, pattern recognition, dimension reduction, abstraction, visualization etc. are directly related to data distribution. A true representation of data enhances the capability of those techniques when applied in real-world applications.

We illustrated that an example of a real world databases (a.k.a. Russian Contract Killing (RCK)) are skewed and exhibit *Zipf* law distribution and do not follow the conventional assumption of Normal distribution. In addition, we compared some of the characteristics of real-world and simulated version of RCK and illustrated that they have different distribution.

This report is a small part of an on going project. The main focus of our effort is to provide a novel framework for LD by combining knowledge representation and reasoning technology with data mining and statistical reasoning. However, as future work of this report, we

would like to do the same analysis for some other possible available real world data such as companies inside information, publication-authors data and dynamic pricing market data such as e-auction to analyze the data characteristics in more details. In addition, we would like to study the effect of such characteristics on design and to evaluate performance of LD techniques which work properly on synthetic data.

Acknowledgments. This work was supported by the Defense Advance Research Projects Agency under Air Force Research Laboratory contract F30602-01-2-0583. The author would like to thank the KOJAK Team: Hans Chalupsky, Tom Russ and Andre Valente for their comments and support.

7. References

- Adamic, L. (2000). Zipf, power-laws, and pareto - a ranking tutorial. http://www.parc.xerox.com/istl/groups/iea/papers/ranking/ranking.html.
- Adibi, J., Shen, W-M. (2001). Self Similar Layered Hidden Markov Model. 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01), Freiburg, Germany.
- Burlaga, F., Klein, L. (1986). "Fractal structure of the interplanetary magnetic." Journal of Geophysical Research 91: 347-350.
- Cook, W., and O'Hayon, G. (2000). "Chronology of Russian killings." Transnational Organized Crime 4(2).
- Dzeroski, S., and Lavrac, N., (2001). Relational Data Mining. Berlin, Springer Verlag.
- Faloutsos, C. (2001). "Tutorial: Next generation data mining tools, using SVD and fractals." http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/christos/www/TALKS/ICDE2001-tut/.
- Fayyad, U., Piatetsky Shapiro, G. P. Smyth, and Uthurusuamy, R. (1996). Advances in knowledge discovery and machine learning. Cambridge, MA, AAAI/MIT Press.
- Feder, J. (1988). Fractals. New York, Plenum Press.
- Hill, B. (1974). "The rank-frequency form of zipf ’s law." Journal of the American Statistical Association, 69(338): 1017–1026.
- Hsu, K. a. H., A. (1990). "Fractal Geometry of Music." The National Academy of Science 87: 938-941.
- Koller, D., and Pfeffer, A (1998). Probabilistic frame-based systems. 16th National Conference on Artifial Intelligence, Madison, WI, AAAI Press / The MIT Press.
- Mandelbrot, B. (1965). "Self-similar error clusters in communications systems and the concept of conditional systems and the concept of conditional stationary." IEEE Transactions on Communications Technology, 13: 71-90.
- Mandelbrot, B. (1982). The Fractal Geometry of Nature. San Francisco, W.H. Freeman and Co.
- Mooney, R., Meville, P., Tang, R., Shavlik, J., Castro Dutra, I., Page, D. and Santos Costa, V. (2002). Relational Data Mining with Inductive Logic Programming for Link Discovery. The National Science Foundation Workshop on Next Generation Data Mining, Baltimore, MD.
- Piatetsky-Shapiro, G., Brachman, R. Khabaza, T. , Kloesgen, W., Simoudis, E. (1996). An overview of issues in developing Industrial data mining and knowledge discovery applications,. Second International Conference on Knowledge Discovery and Data Mining.
- Schroeder, M. (1991). Fractals, Chaos, Power Laws. New York:, W.H. Freeman and Company,.
- Senator, T. (2001). "Evidence Extraction and Link Discovery."
- Team, E. P. E., Lead, I. P. E., and Powers, J. (2002). Taskbased simulator version, Information Extraction and Transport, Inc.
- Williams, P. (2002). Patterns, indicators, and warnings in link analysis: The contract killings dataset., Veridian Systems Division.
- Willinger, W., Taqqu M. S., Sherman, W. and Wilson, D. (1997). "Self-similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level." IEEE/ACM Transactions on Networking 5(1): 71-86.
- Zheng, Z., Kohavi, R. and Mason, L. (2001). Real World Performance of Association Rule Algorithms. The Seventh International Conference on Knowledge Discovery and Data Mining.
- Zhiqiang, B., Christos F.and Flip K. (2001). The "DGX" Distribution for Mining Massive, Skewed Data. The Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA.